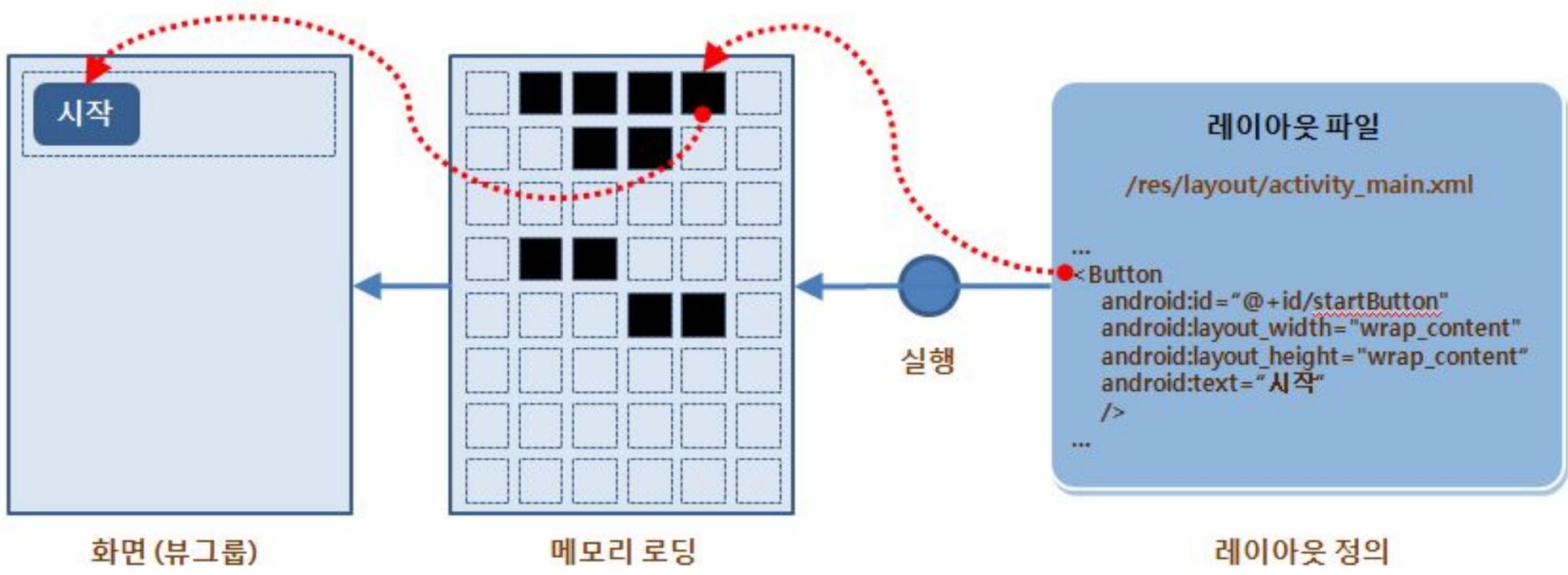


1.

레이아웃 인플레이션

인플레이션이란?

- 인플레이션 : XML 레이아웃에 정의된 내용이 메모리에 객체화되는 과정

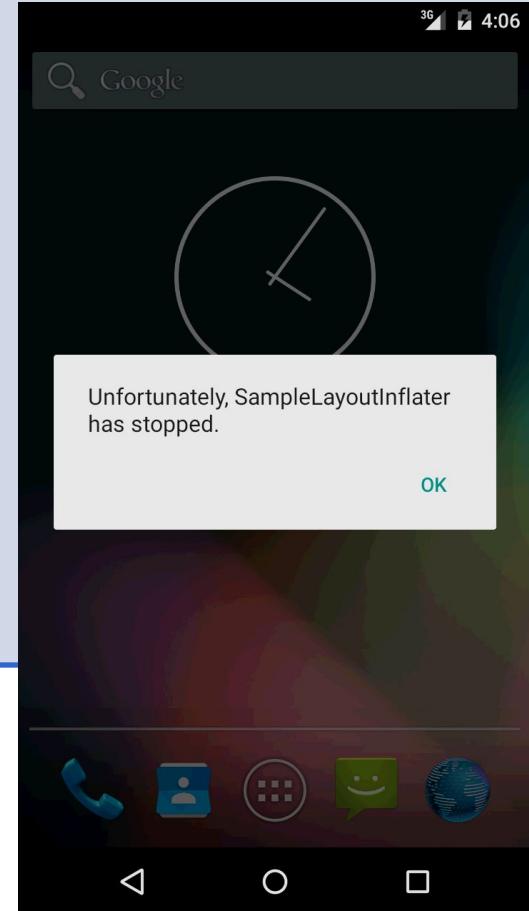


[“시작” 버튼의 레이아웃 인플레이션 과정]

레이아웃 인플레이션의 이해 – 호출 순서

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        Button button1 = (Button) findViewById(R.id.button1);  
        button1.setText("시작됨");  
  
        setContentView(R.layout.activity_main);  
    }  
}
```

[`setContentView()` 코드와 `findViewById()` 메소드의 호출 순서를 바꾼 경우]



setContentView() 메소드의 역할

[Reference]

```
public void setContentView (int layoutResID)  
public void setContentView (View view [, ViewGroup.LayoutParams params])
```

- **setContentView() 메소드의 역할**

- 화면에 나타낼 뷰를 지정하는 역할
- XML 레이아웃의 내용을 메모리 상에 객체화하는 역할

[Reference]

```
getSystemService(Context.LAYOUT_INFLATER_SERVICE)
```

- 전체 화면 중에서 일부분만을 차지하는 화면 구성요소들을 XML 레이아웃에서 로딩하여 보여줄 수 없을까?
- LayoutInflator라는 클래스를 제공하며, 이 클래스는 시스템 서비스로 제공됨

2.

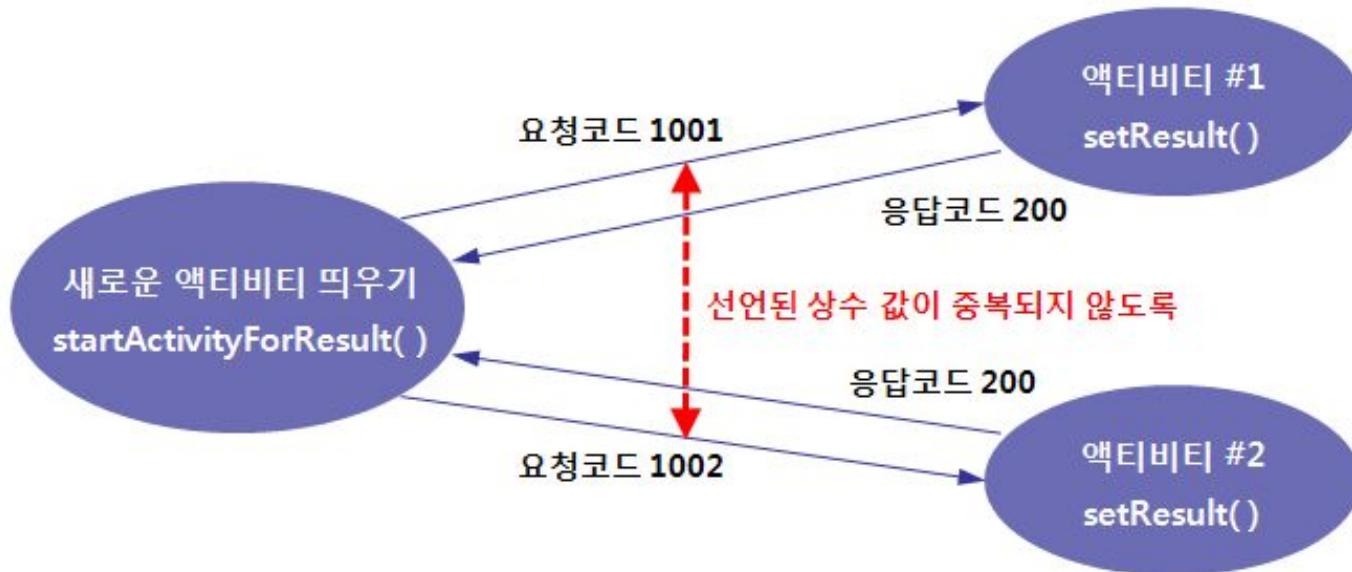
화면 구성과 화면간 이동

액티비티



[안드로이드 애플리케이션을 구성하는 네 가지 구성요소]

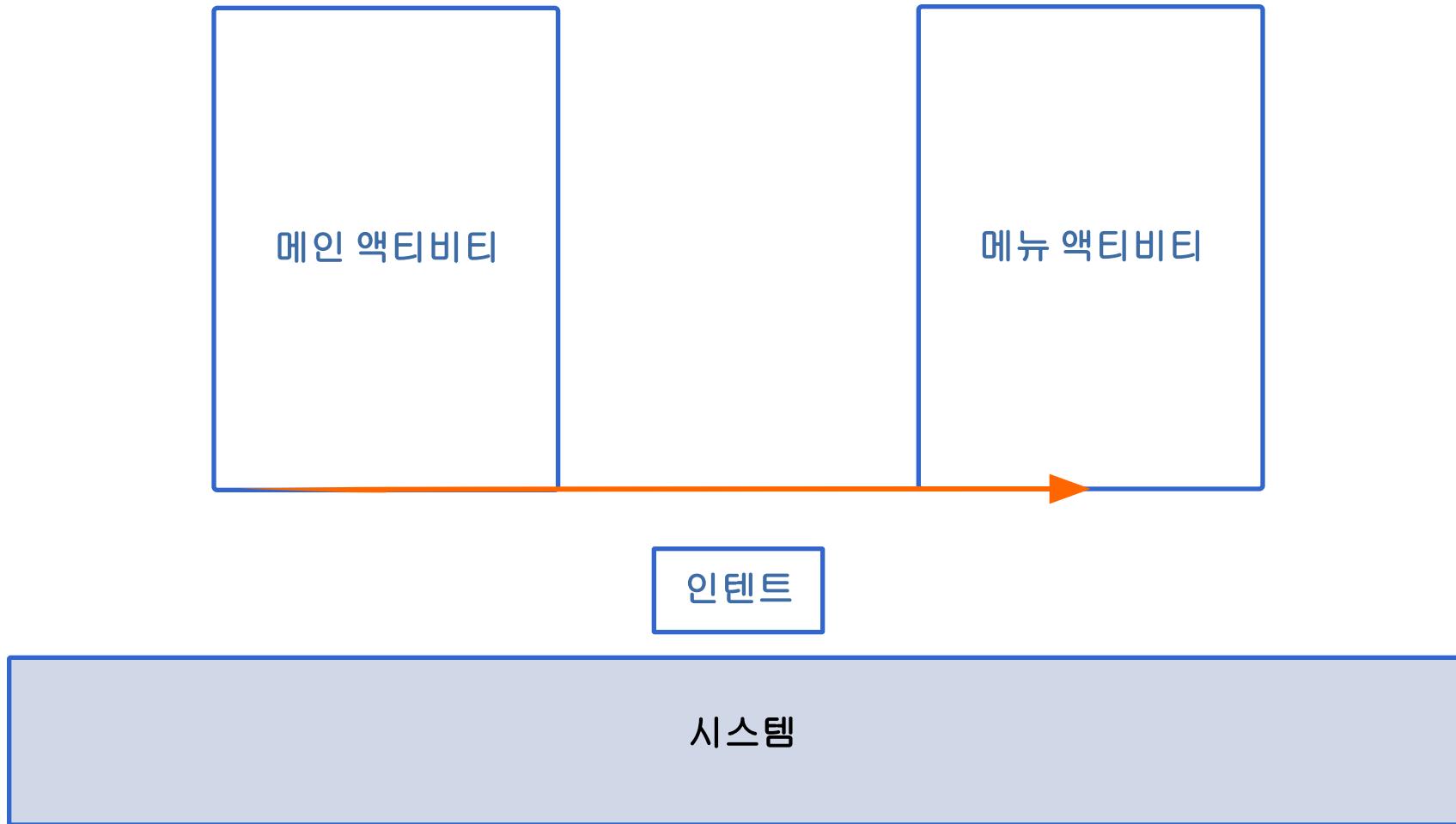
화면 구성과 화면간 이동 과정



[Reference]

```
protected void onActivityResult(int requestCode, int resultCode, Intent Data)
```

화면을 띄울 때 시스템으로 요청하기



액티비티 만들기 예제

액티비티 만들기 예제

- 또 다른 액티비티를 메인 액티비티에서 띄워주기
- 또 다른 액티비티 정의하기

메인 액티비티 코드에서
액티비티 띄우기

- 인텐트 생성과 액티비티 띄우기

또 다른 액티비티 정의

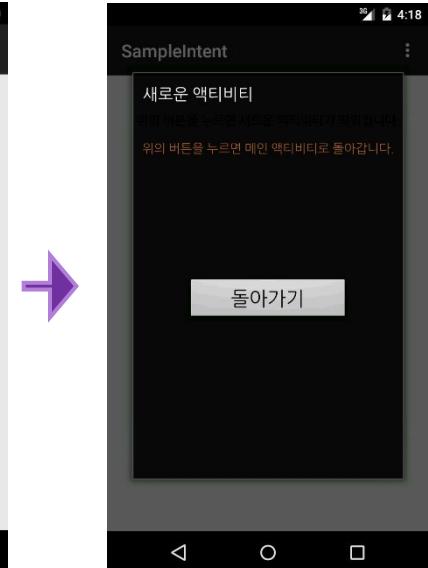
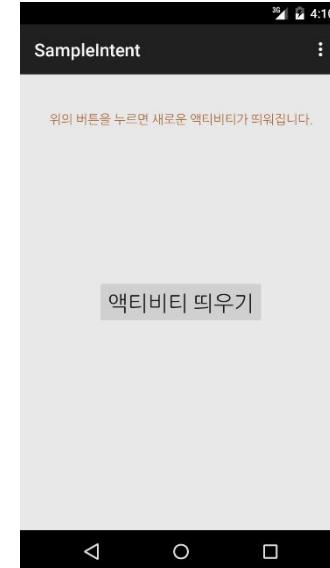
- 또 다른 액티비티의 레이아웃과 코드 작성

메인 액티비티 코드에서
응답 처리

- 또 다른 액티비티로부터의 응답
처리

매니페스트에 추가

- 새로 만든 액티비티를 매니페스트에 추가



메인 액티비티에서 새로운 화면 띄우기

[Reference]

`startActivityForResult(Intent intent, int requestCode)`

```
public class MainActivity extends AppCompatActivity {  
    public static final int REQUEST_CODE_ANOTHER = 1001; 1 다른 액티비티를 띄우기 위한 요청코드 정의  
    ...  
    public void onCreate(Bundle savedInstanceState) {  
        ...  
        Button startBtn = (Button) findViewById(R.id.startBtn);  
        startBtn.setOnClickListener(new OnClickListener() {  
            public void onClick(View v) {  
                Intent intent = new Intent(getApplicationContext(), AnotherActivity.class );  
                2 또 다른 액티비티를 띄우기 위한 인텐트 객체 생성  
                startActivityForResult(intent, REQUEST_CODE_ANOTHER ); 3 액티비티 띄우기  
            }  
        });  
    }  
}
```

새로운 액티비티로부터의 응답 처리

```
...
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE_ANOTHER) {
        Toast toast = Toast.makeText(getApplicationContext(),
            "onActivityResult called with code : " + resultCode,
            Toast.LENGTH_LONG);
        toast.show();
    }
}
}

if (resultCode == 1) {
    String name = data.getExtras().getString("name");
    toast = Toast.makeText(getApplicationContext(), "result name : " + name, Toast.LENGTH_LONG);
    toast.show();
}
```

4 또 다른 액티비티에서 보내온 응답인지 요청코드로 확인

5 토스트로 메시지 보이기

또 다른 액티비티의 자바 코드

```
...
public class AnotherActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.another);

        Button returnBtn = (Button) findViewById(R.id.returnBtn );
        returnBtn.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Intent resultIntent = new Intent();
                resultIntent.putExtra("name", "mike");
                setResult(1, resultIntent);
                finish();
            }
        });
    }
}
```

1 버튼 객체 참조

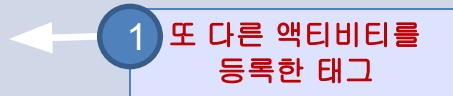
2 인텐트 객체 생성하고 name의 값을 부가 데이터로 넣기

3 응답 보내기

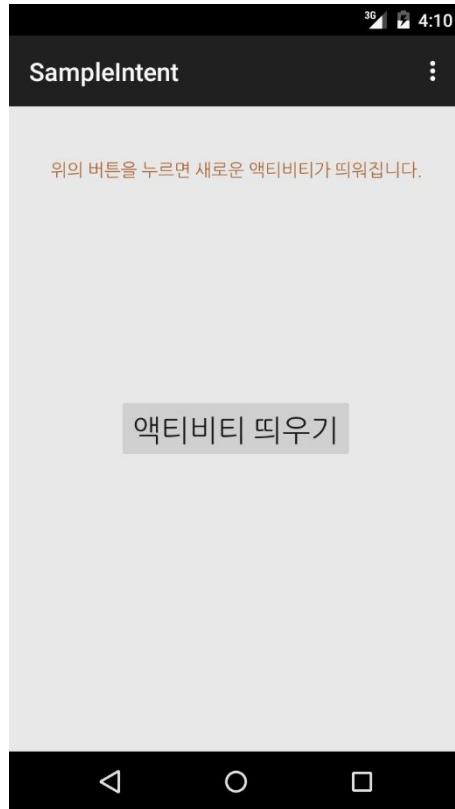
4 이 액티비티 없애기

매니페스트에 액티비티 태그 추가

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0" package="org.androidtown.helloworld">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
            android:label="@string/app_name">
        </activity>
        <activity android:name=".AnotherActivity"
            android:label="@string/app_name">
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```



애플리케이션 실행 화면



액티비티 구성 과정 정리

(1) 새로운 액티비티의 XML 레이아웃 정의

(2) 새로운 액티비티 코드 작성

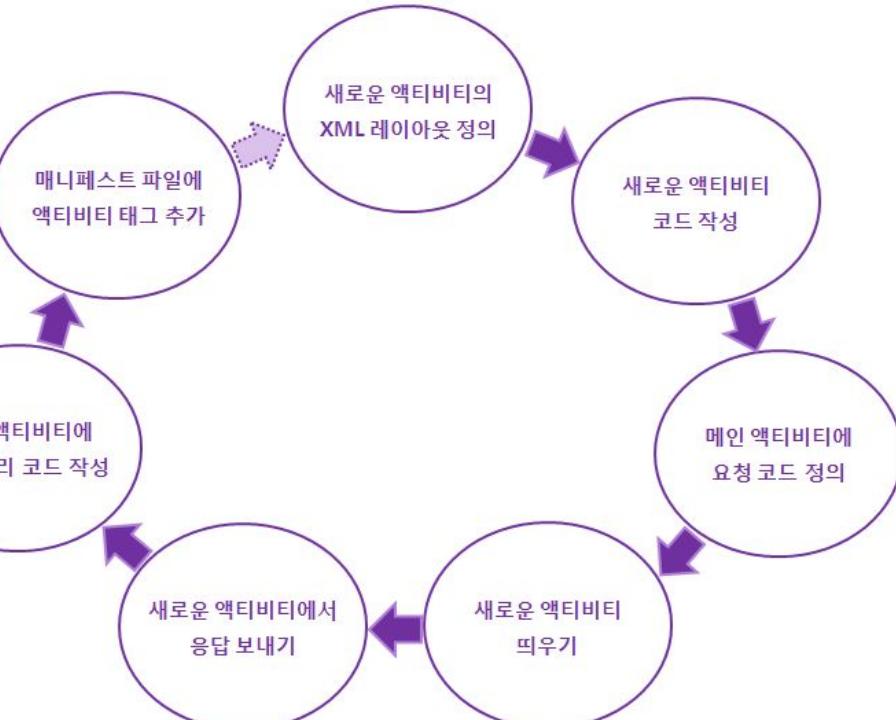
(3) 메인 액티비티에 요청 코드 정의

(4) 새로운 액티비티 띄우기

(5) 새로운 액티비티에서 응답 보내기

(6) 메인 액티비티에 응답 처리 코드 작성

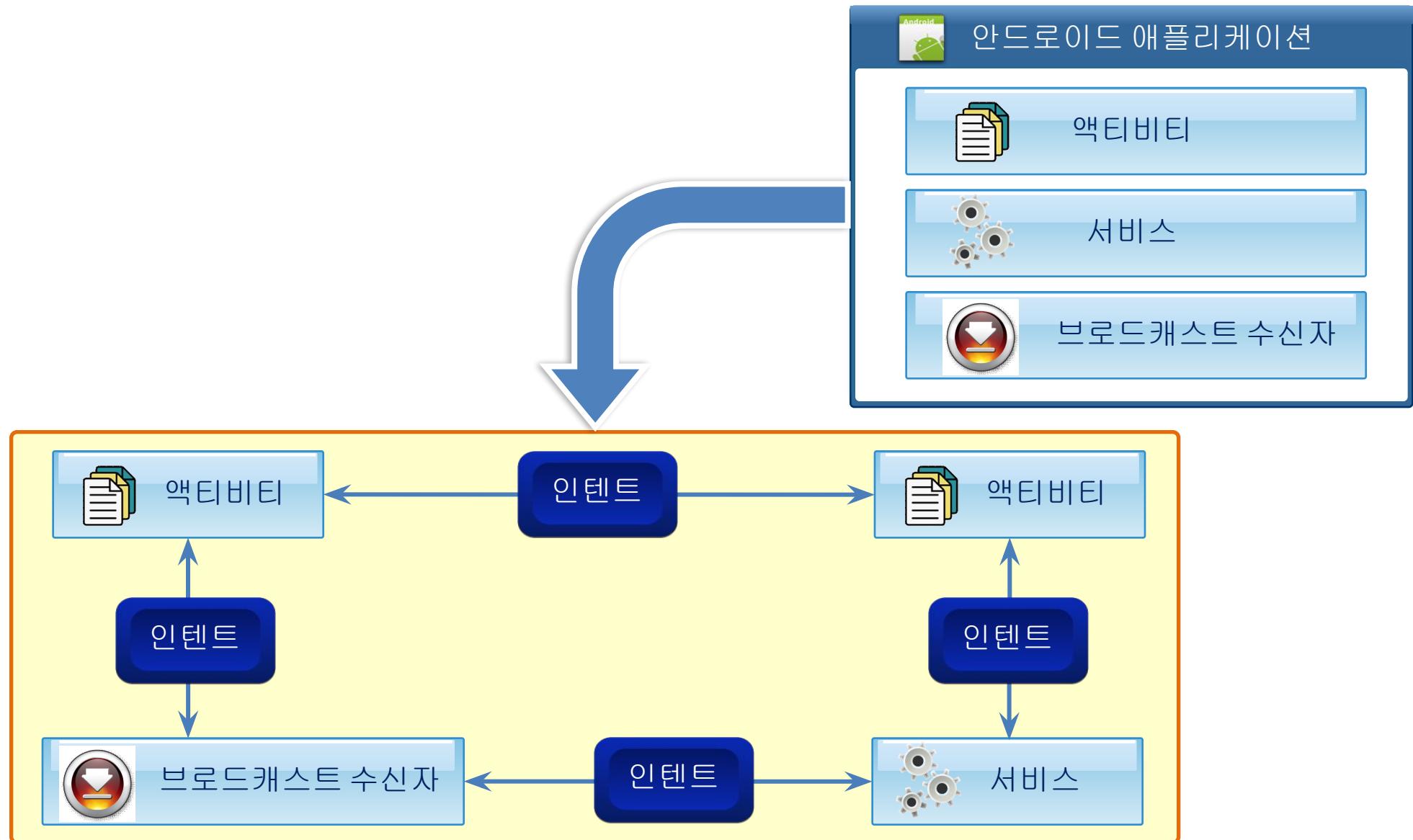
(7) 매니페스트 파일에 액티비티 태그 추가



3.

인텐트와 데이터 전달

인텐트와 데이터 전달



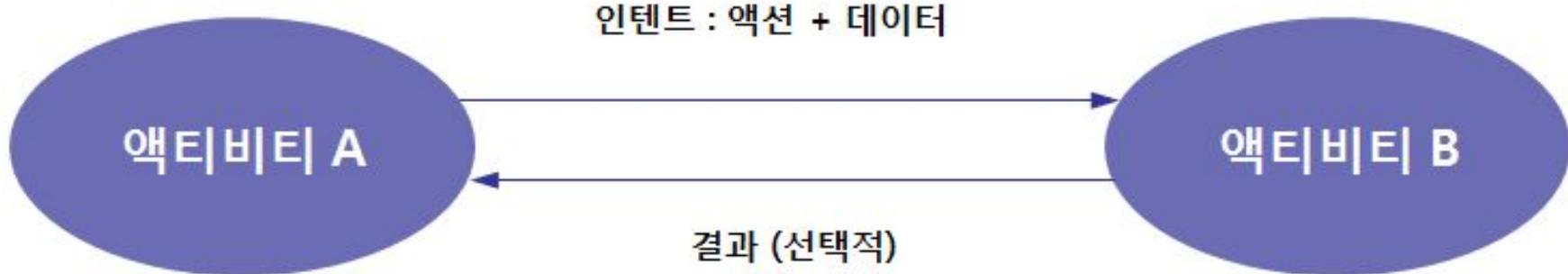
액티비티 간의 인텐트 전달

[Reference]

`startActivity()`

`startService()` 또는 `bindService()`

`broadcastIntent()`



[액티비티 간의 인텐트 전달]

액션과 데이터를 사용하는 대표적인 예

속성	설명
ACTION_DIAL tel:01077881234	주어진 전화번호를 이용해 전화걸기 화면을 보여줌
ACTION_VIEW tel:01077881234	주어진 전화번호를 이용해 전화걸기 화면을 보여줌. URI 값의 유형에 따라 VIEW 액션이 다른 기능을 수행함
ACTION_EDIT content://contacts/people/2	전화번호부 데이터베이스에 있는 정보 중에서 ID 값이 2인 정보를 편집하기 위한 화면을 보여줌
ACTION_VIEW content://contacts/people	전화번호부 데이터베이스의 내용을 보여줌

명시적 인텐트와 암시적 인텐트

[Reference]

`Intent()`

`Intent(Intent o)`

`Intent(String action [,Uri uri])`

`Intent(Context packageContext, Class<?> cls)`

`Intent(String action, Uri uri, Context packageContext, Class<?> cls)`

- 명시적 인텐트(**Explicit Intent**)

- 인텐트에 클래스 객체나 컴포넌트 이름을 지정하여 호출할 대상을 확실히 알 수 있는 경우

- 암시적 인텐트(**Implicit Intent**)

- 액션과 데이터를 지정하긴 했지만 호출할 대상이 달라질 수 있는 경우

- 범주(category), 타입(Type), 컴포넌트(component), 부가 데이터(extras)

인텐트의 대표적 속성

- 범주 (Category)

- 액션이 실행되는 데 필요한 추가적인 정보를 제공

- 타입 (Type)

- 인텐트에 들어가는 데이터의 MIME 타입을 명시적으로 지정

- 컴포넌트 (Component)

- 인텐트에 사용될 컴포넌트 클래스 이름을 명시적으로 지정

- 부가 데이터 (Extra)

- 인텐트는 추가적인 정보를 넣을 수 있도록 번들(Bundle) 객체를 담고 있음
- 이 객체를 통해 인텐트 안에 더 많은 정보를 넣어 다른 애플리케이션 구성요소에 전달할 수 있음

데이터 전달 예제

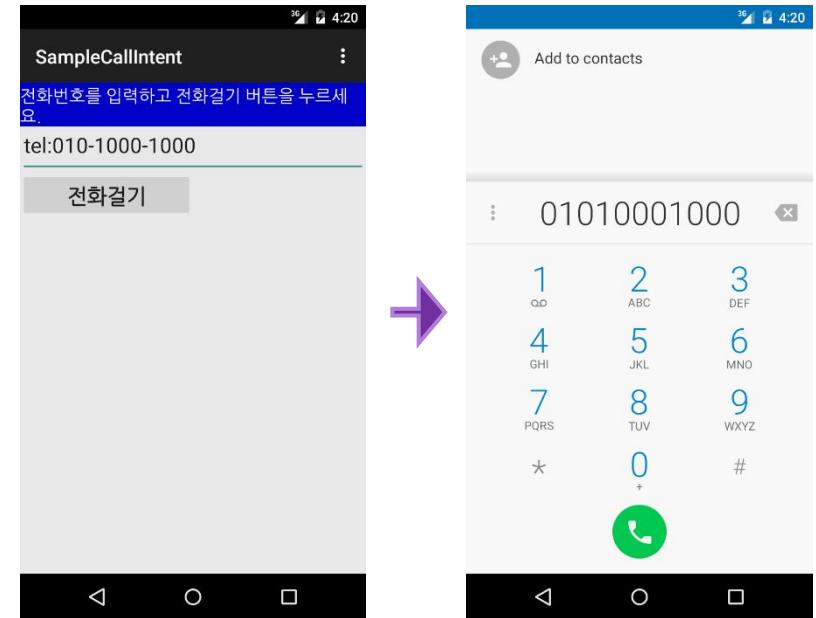
데이터 전달 예제

- 액티비티 간에 인텐트로 데이터 전달
- 전화걸기 기능 이용

XML 레이아웃 정의

- 전화걸기 화면을 띄우는 화면 구성
- 전화걸기 화면을 띄울 때 전화번호 전달

메인 액티비티 코드 작성



XML 레이아웃 만들기

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    >  
    <TextView  
        android:id="@+id/text01"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:background="#ff0000cc"  
        android:text="전화 걸기 인텐트"  
        android:textSize="20dp"  
    />
```

1

텍스트뷰 정의

XML 레이아웃 만들기 (계속)

```
<EditText  
    android:id="@+id/edit01"  
    android:layout_width="match_parent"  
    android:layout_height="36dp"  
    android:text="tel:010-7788-1234"  
    android:textSize="18dp"  
/>  
<Button  
    android:id="@+id	btnCall"  
    android:layout_width="80dp"  
    android:layout_height="wrap_content"  
    android:text="전화걸기"  
    android:textSize="18dp"  
    android:textStyle="bold"  
/>  
</LinearLayout>
```

2 입력상자 정의

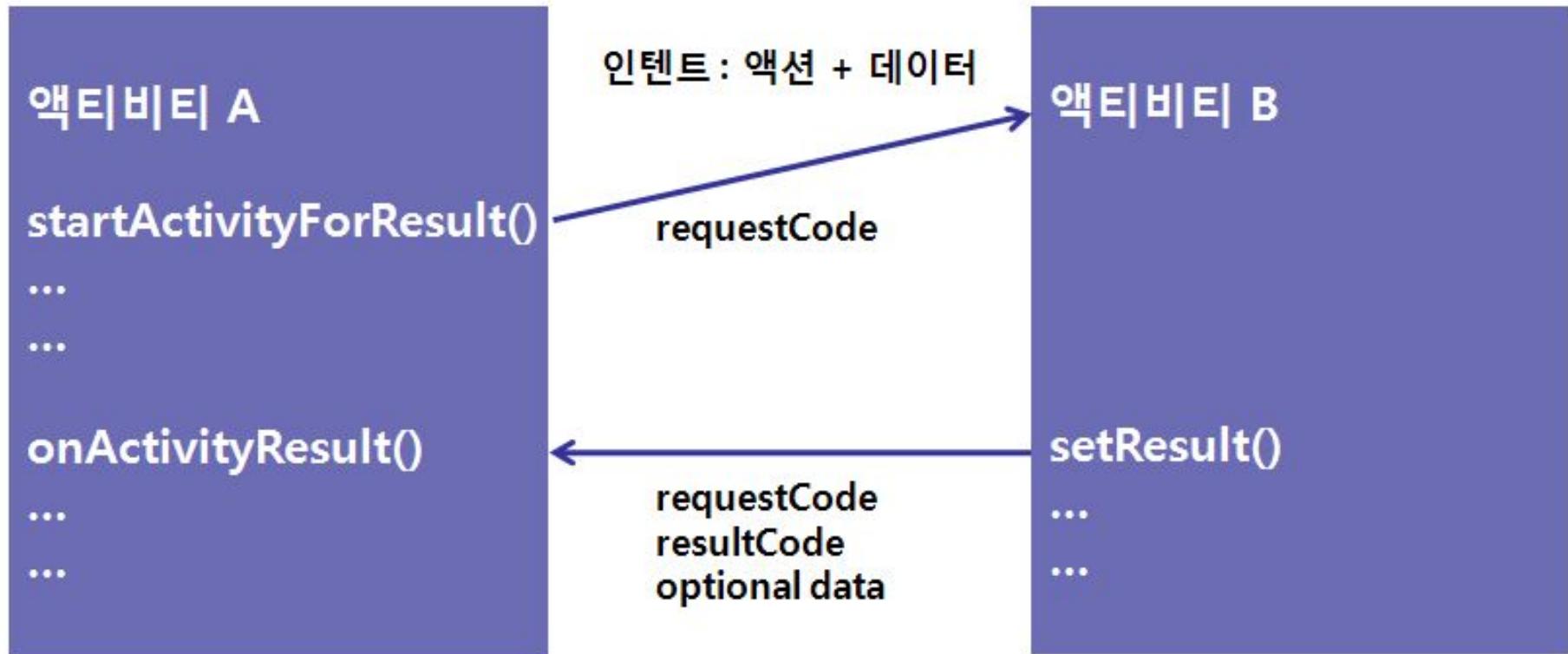
3 버튼 정의

메인 액티비티 코드 만들기

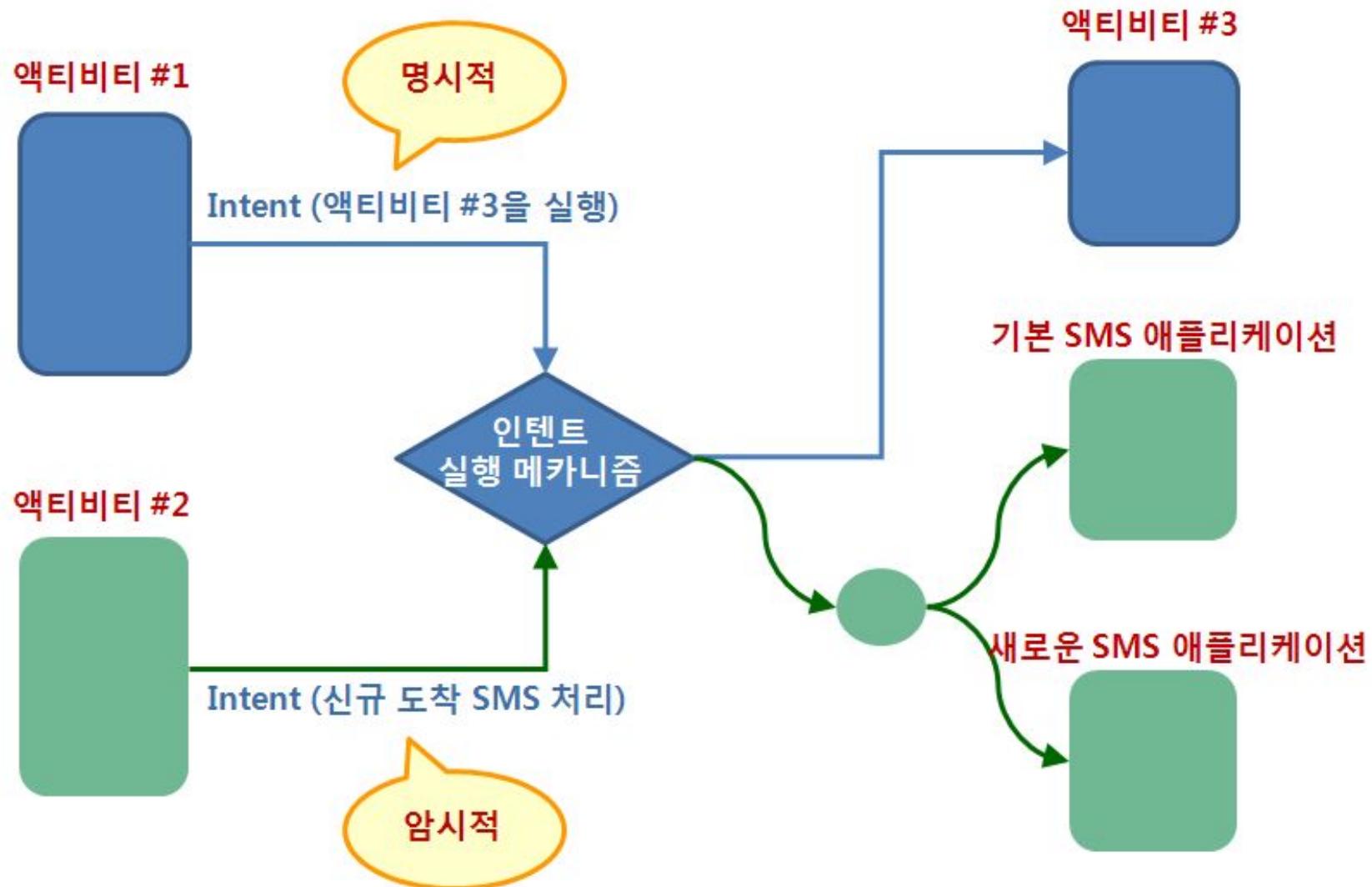
```
Intent myActivity2 = new Intent(Intent.ACTION_DIAL, Uri.parse(myData ));  
startActivity(myActivity2);
```

```
Intent intent = new Intent();  
ComponentName name = new ComponentName("org.androidtown.intent.basic",  
        "org.androidtown.intent.basic.AnotherActivity" );  
intent.setComponent(name);  
startActivityForResult(intent, REQUEST_CODE_ANOTHER );
```

액티비티 간의 데이터 전달 방법 정리



인텐트를 해석하는 과정



PDF 문서 보기 예제

PDF 문서 보기 예제

- 인텐트를 이용해 PDF 문서 보기
- 파일명을 데이터로 전달하여 PDF 문서 열기

XML 레이아웃 정의

- PDF 보기 위한 화면 구성

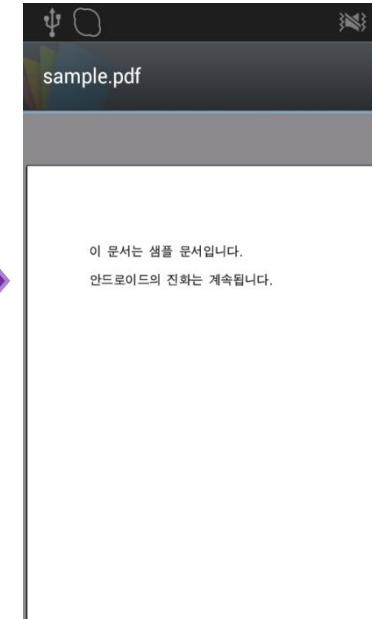
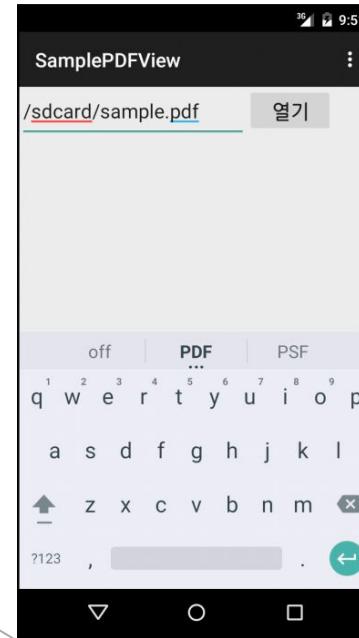
메인 액티비티 코드 작성

- 인텐트를 이용해 PDF를 보기 위한 코드 구성

샘플 PDF 파일 복사

- 단말에 샘플 PDF 파일 복사

3. 인텐트와 데이터 전달



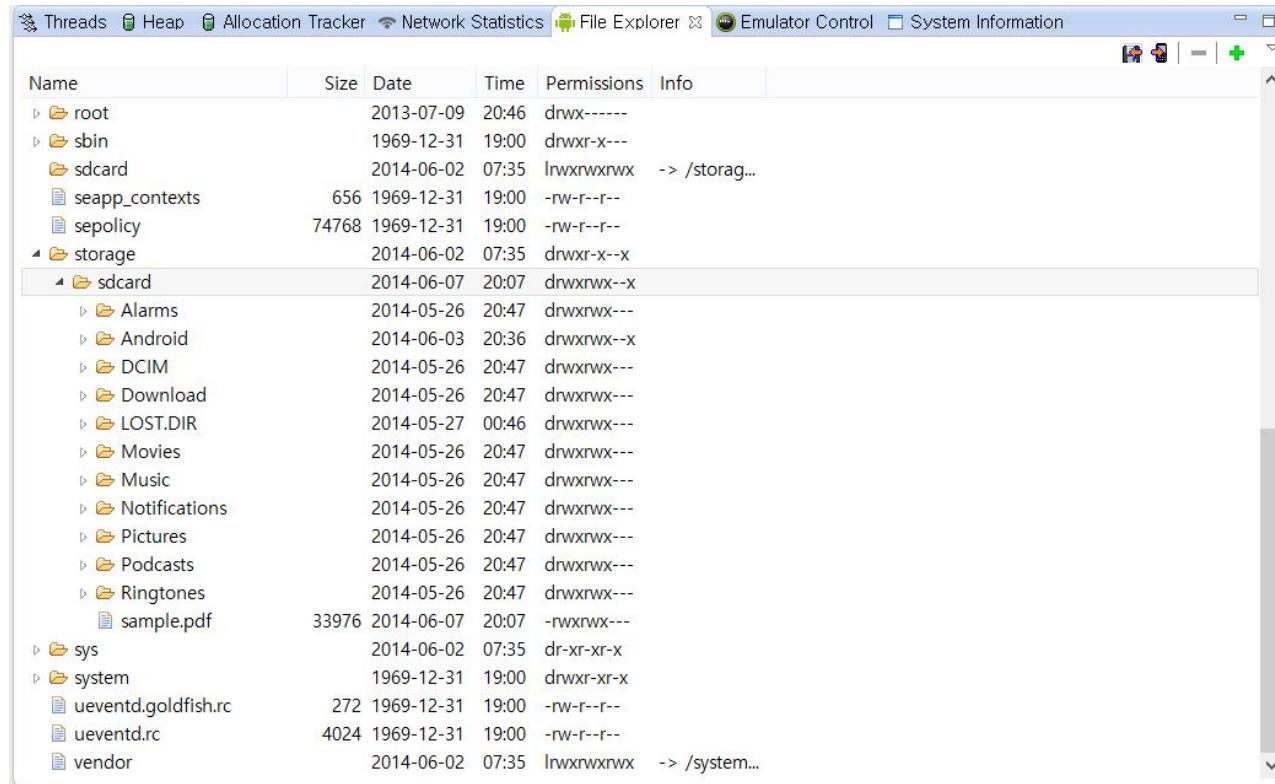
메인 액티비티 코드 만들기

```
public void openPDF(String contentsPath) {  
    File file = new File(contentsPath);  
    if (file.exists()) {  
        Uri path = Uri.fromFile (file);  
        Intent intent = new Intent(Intent.ACTION_VIEW );  
        intent.setDataAndType(path, "application/pdf" );  
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP );  
        try {  
            startActivity(intent);  
        } catch (ActivityNotFoundException e) {  
            Toast.makeText(this,  
                    "No Application Available to View PDF",  
                    Toast.LENGTH_SHORT).show();  
        }  
    } else {  
        Toast.makeText(this, "PDF 파일이 없습니다.", Toast.LENGTH_SHORT).show();  
    }  
}
```

3 PDF 파일 열기 기능을 정의한 메소드
4 Uri 객체로 생성
5 ACTION_VIEW 액션을 가지는 인텐트 객체 생성
6 Uri 객체와 MIME 타입 지정
7 플래그 설정
8 액티비티 띄우기

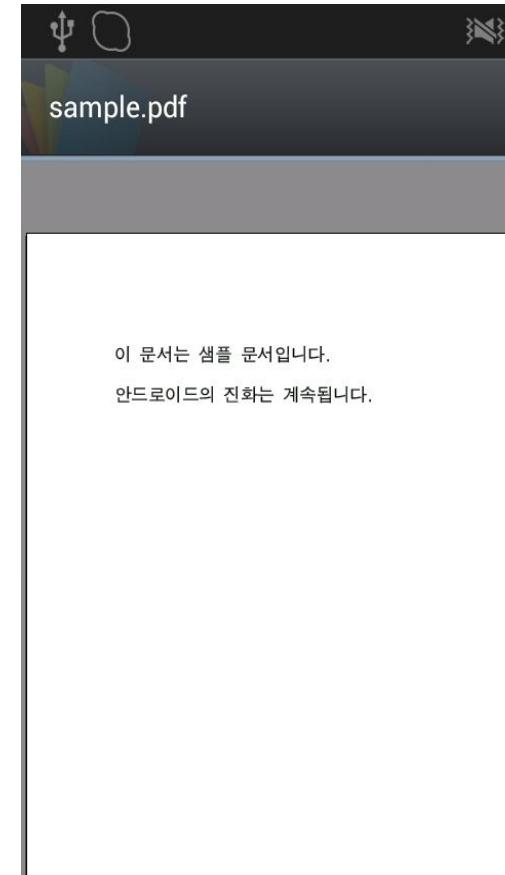
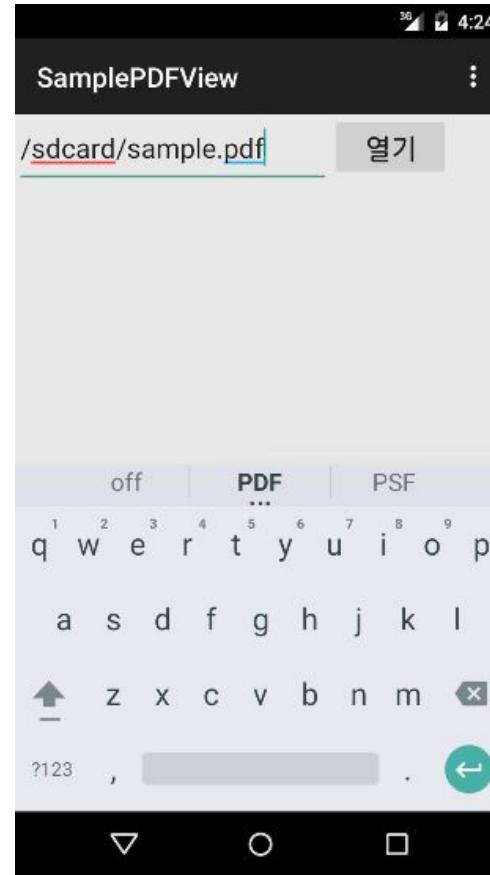
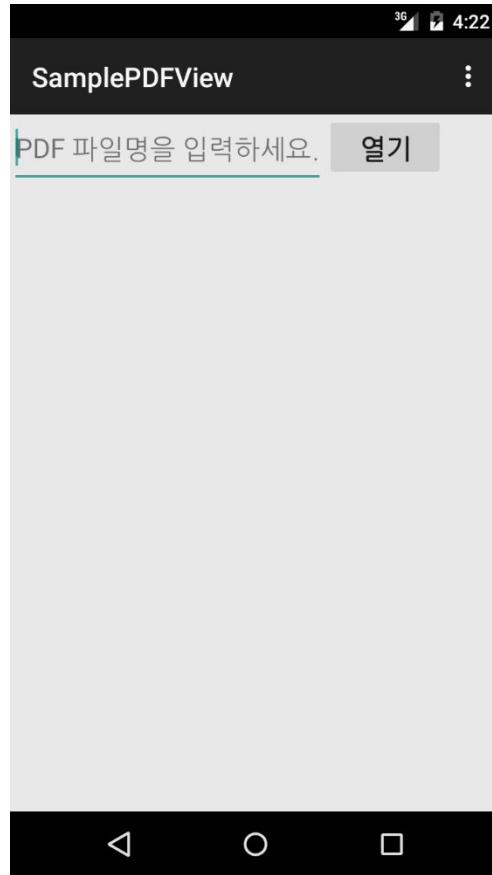
샘플 PDF 파일을 단말로 복사하기

- 단말에 연결한 후 샘플 PDF 파일 복사
- DDMS 창의 Emulator 탭에서 단말 선택
- [File Explorer] 탭의 우측 상단에 있는
- [전송] 버튼을 누름

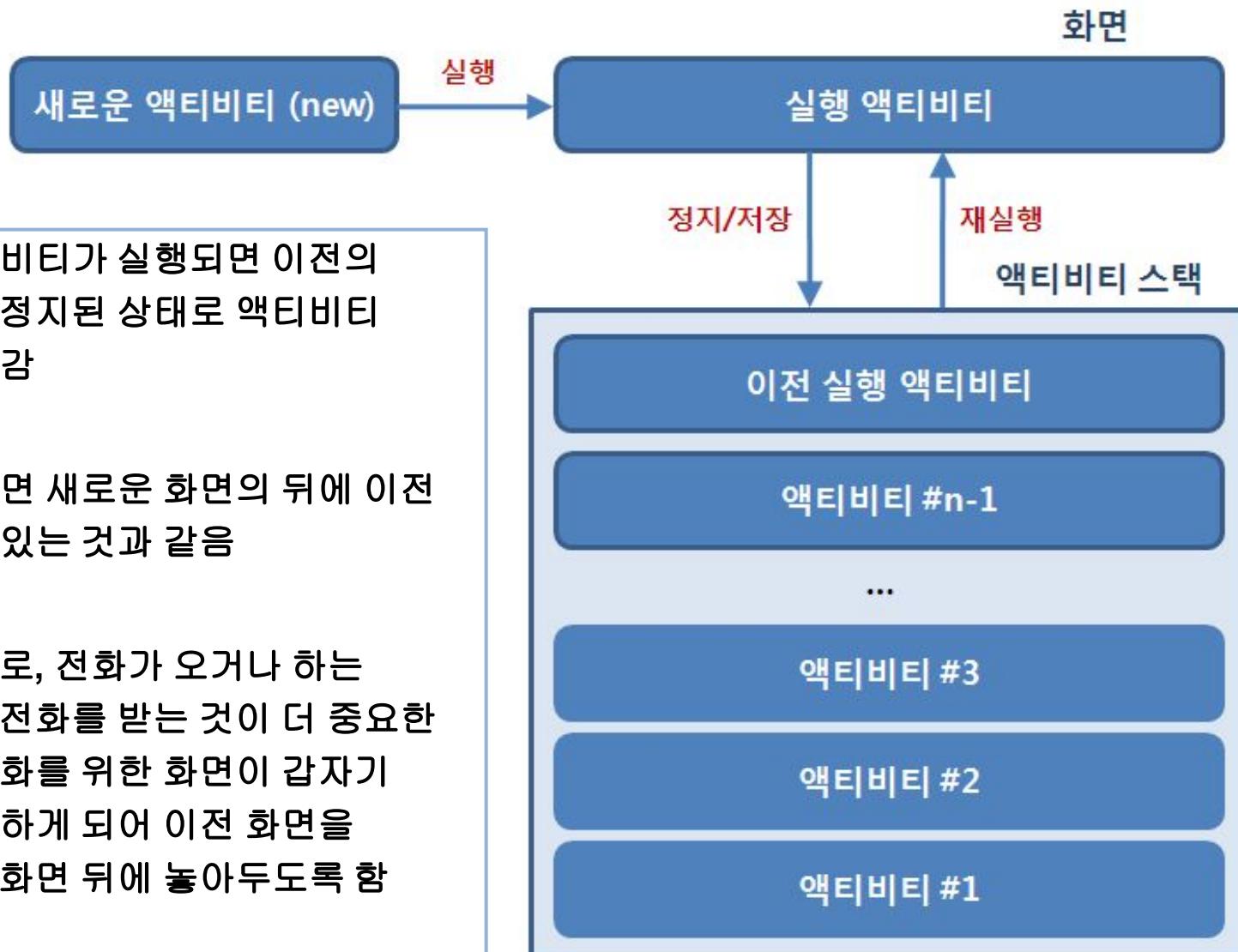


3. 인텐트와 데이터 전달

PDF 문서보기 실행 화면



액티비티 스택



액티비티 스택과 플래그 사용

[Reference]

FLAG_ACTIVITY_SINGLE_TOP
FLAG_ACTIVITY_NO_HISTORY
FLAG_ACTIVITY_CLEAR_TOP

- 새로운 액티비티를 실행할 때마다 메모리에 새로운 객체를 만들고 이전 화면 위에 쌓는 방식은 비효율적일 수 있음
- 동일한 화면이 이미 만들어져 있는 경우에는 그 화면을 그대로 보여주고 싶다면 플래그를 사용하면 됨

NO_FLAG



FLAG_ACTIVITY_SINGLE_TOP



[**FLAG_ACTIVITY_SINGLE_TOP** 플래그를 사용한 경우]

액티비티 플래그 사용 예

```
Intent intent = new Intent(getApplicationContext(), AnotherActivity.class );
intent.putExtra("startCount", String.valueOf(startCount));
intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP );
startActivityForResult(intent, REQUEST_CODE_ANOTHER );
```

- 1 인텐트 객체 생성
- 2 부가 데이터 넣기
- 3 인텐트 플래그 설정
- 4 인텐트 띄우기

다른 액티비티 플래그의 사용

NO_FLAG



FLAG_ACTIVITY_NO_HISTORY



[FLAG_ACTIVITY_NO_HISTORY 플래그를 사용한 경우]

NO_FLAG



FLAG_ACTIVITY_CLEAR_TOP



[FLAG_ACTIVITY_CLEAR_TOP 플래그를 사용한 경우]

4.

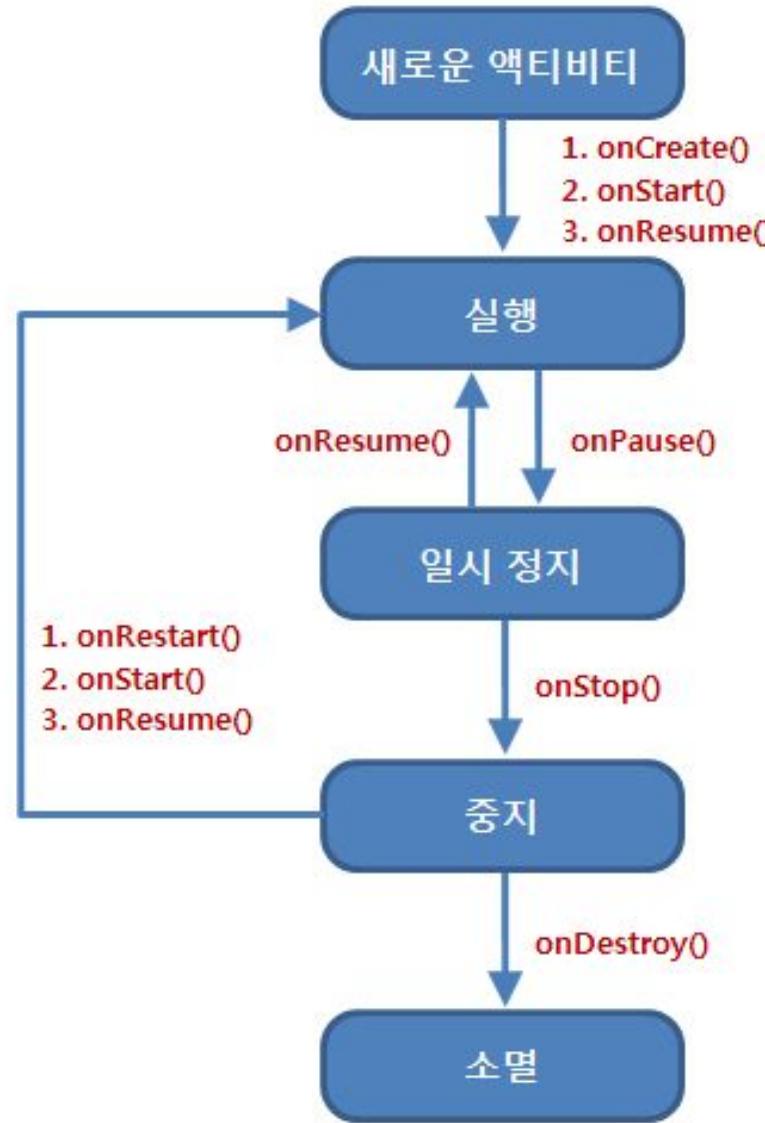
수명주기

수명 주기

상태	설명
실행(Running)	화면 상에 액티비티가 보이면서 실행되어 있는 상태. 액티비티 스택의 최상위에 있으며 포커스를 가지고 있음
일시 중지(Paused)	사용자에게 보이기는 하지만 다른 액티비티가 위에 있어 포커스를 받지 못하는 상태. 대화상자가 위에 있어 일부가 가려져 있는 경우에 해당함
중지(Stopped)	다른 액티비티에 의해 완전히 가려져 보이지 않는 상태

[액티비티의 대표적인 상태 정보]

수명주기에 따른 상태 변화



액티비티의 상태 메소드

상태 메소드	설명
onCreate()	<ul style="list-style-type: none">- 액티비티가 처음에 만들어졌을 때 호출됨- 화면에 보이는 뷰들의 일반적인 상태를 설정하는 부분- 이전 상태가 저장되어 있는 경우에는 번들 객체를 참조하여 이전 상태 복원 가능- 이 메소드 다음에는 항상 onStart() 메소드가 호출됨
onStart()	<ul style="list-style-type: none">- 액티비티가 화면에 보이기 바로 전에 호출됨- 액티비티가 화면 상에 보이면 이 메소드 다음에 onResume() 메소드가 호출됨- 액티비티가 화면에서 가려지게 되면 이 메소드 다음에 onStop() 메소드가 호출됨
onResume()	<ul style="list-style-type: none">- 액티비티가 사용자와 상호작용하기 바로 전에 호출됨
onRestart()	<ul style="list-style-type: none">- 액티비티가 중지된 이후에 호출되는 메소드로 다시 시작되기 바로 전에 호출됨- 이 메소드 다음에는 항상 onStart() 메소드가 호출됨
onPause()	<ul style="list-style-type: none">- 또 다른 액티비티를 시작하려고 할 때 호출됨- 저장되지 않은 데이터를 저장소에 저장하거나 애니메이션 중인 작업을 중지하는 등의 기능을 수행하는 메소드임- 이 메소드가 리턴하기 전에는 다음 액티비티가 시작될 수 없으므로 이 작업은 매우 빨리 수행된 후 리턴되어야 함- 액티비티가 이 상태에 들어가면 시스템은 액티비티를 강제 종료할 수 있음
onStop()	<ul style="list-style-type: none">- 액티비티가 사용자에게 더 이상 보이지 않을 때 호출됨- 액티비티가 소멸되거나 또 다른 액티비티가 화면을 가릴 때 호출됨- 액티비티가 이 상태에 들어가면 시스템은 액티비티를 강제 종료할 수 있음
onDestroy()	<ul style="list-style-type: none">- 액티비티가 소멸되어 없어지기 전에 호출됨- 이 메소드는 액티비티가 받는 마지막 호출이 됨- 액티비티가 애플리케이션에 의해 종료되거나 (finish() 메소드 호출) 시스템이 강제로 종료시키는 경우에 호출될 수 있음- 위의 두 가지 경우를 구분할 때 isFinishing() 메소드를 이용함- 액티비티가 이 상태에 들어가면 시스템은 액티비티를 강제 종료할 수 있음

수명주기 확인하기 예제

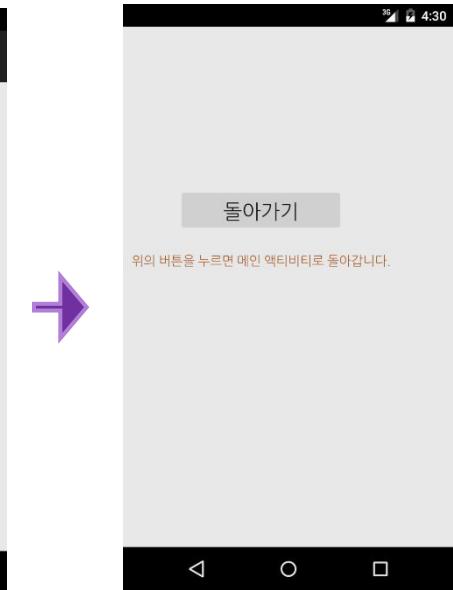
수명주기 확인하기 예제

- 액티비티 상태에 따른 수명주기 확인하기
- 상태 메소드 별로 토스트 메시지 추가

XML 레이아웃 정의

- 입력상자와 버튼이 있는 레이아웃
- 상태 메소드 별로 토스트 메시지 코드 추가

메인 액티비티 코드 작성



메인 액티비티 코드 만들기 (계속)

```
@Override  
protected void onDestroy() {  
    super.onDestroy();  
    Toast.makeText(getApplicationContext(), "onDestroy ...", Toast.LENGTH_LONG).show();  
}  
  
@Override  
protected void onPause() {  
    super.onPause();  
    saveCurrentState(); ← 4 현재 상태 저장  
    Toast.makeText(getApplicationContext(), "onPause ...", Toast.LENGTH_LONG).show();  
}  
  
@Override  
protected void onRestart() {  
    super.onRestart();  
    Toast.makeText(getApplicationContext(), "onRestart ...", Toast.LENGTH_LONG).show();  
}
```

Continued..

메인 액티비티 코드 만들기 (계속)

```
@Override  
protected void onResume() {  
    super.onResume();  
    restoreFromSavedState();  
    Toast.makeText(getApplicationContext(), "onResume...", Toast.LENGTH_LONG).show();  
}  
  
@Override  
protected void onStart() {  
    super.onStart();  
    Toast.makeText(getApplicationContext(), "onStart ... ", Toast.LENGTH_LONG).show();  
}  
  
@Override  
protected void onStop() {  
    super.onStop();  
    Toast.makeText(getApplicationContext(), "onStop ... ", Toast.LENGTH_LONG).show();  
}
```

5

현재 상태 복원

Continued..

메인 액티비티 코드 만들기 (계속)

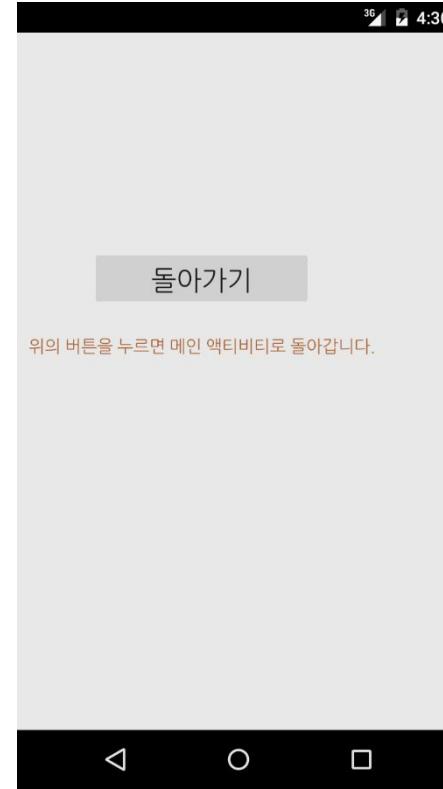
```
protected void restoreFromSavedState() {  
    SharedPreferences myPrefs = getSharedPreferences(PREF_ID, actMode);  
    if ((myPrefs != null) && (myPrefs.contains("txtMsg"))){  
        String myData = myPrefs.getString("txtMsg", "");  
        txtMsg.setText(myData);  
    }  
}  
  
protected void saveCurrentState() {  
    SharedPreferences myPrefs = getSharedPreferences(PREF_ID, actMode);  
    SharedPreferences.Editor myEditor = myPrefs.edit();  
    myEditor.putString( "txtMsg", txtMsg.getText().toString() );  
    myEditor.commit();  
}
```

6 설정 정보에 저장한 상태 값을
읽어 와서 복원

7 상태 값을 설정 정보에 저장

Continued..

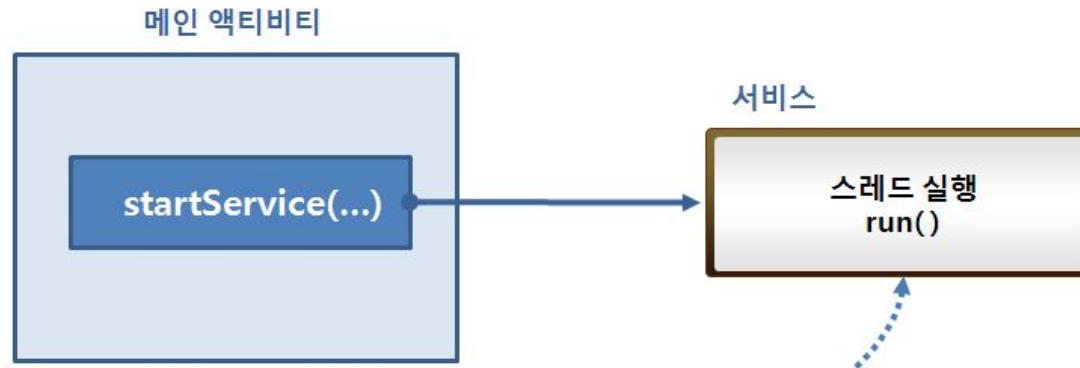
수명주기 확인하기 실행 화면



5.

서비스

서비스



- 서비스는 **백그라운드**에서 실행되는 애플리케이션 구성 요소
- 서비스는 매니페스트 파일(AndroidManifest.xml) 안에 **<service>** 태그를 이용하여 선언
- 서비스를 시작/중지시키는 메소드
 - Context.startService()
 - Context.bindService()
 - stopService(...)
 - unbindService(...)
- 서비스는 다른 구성 요소들처럼 메인 스레드에서 동작
따라서 CPU를 많이 쓰거나 대기 상태(blocking)를 필요로 하는 작업들은 **스레드를 새로** 만들어 주어야 함

서비스 실행 예제

서비스 예제

- 액티비티 상태에 따른 수명주기 확인하기
- 상태 메소드 별로 토스트 메시지 추가

메인 액티비티 코드 작성

- 일정 시간간격으로 메시지를 보여주는 서비스 클래스 정의
- 메인 액티비티에서 서비스 시작

매니페스트에 추가

5. 서비스-새로운 서비스를 매니페스트에 추가

메인 액티비티

startService(...)

서비스

스레드 실행
run()



매니페스트 파일
(AndroidManifest.xml)

서비스 클래스 정의

```
package org.androidtown.basic.service;  
import android.app.Service;  
import android.content.Intent;  
import android.os.IBinder;  
import android.util.Log;  
  
public class MyService extends Service implements Runnable {  
    public static final String TAG = "MyService";  
    private int count = 0;  
    public void onCreate() {  
        super.onCreate();  
        Thread myThread = new Thread(this);  
        myThread.start();  
    }  
}
```

1

시작

Continued..

서비스 클래스 정의 (계속)

```
public void run() {  
    while(true) {  
        try {  
            Log.i(TAG, "my service called #" + count);  
            count++;  
            Thread.sleep(5000);  
        } catch(Exception ex) {  
            Log.e(TAG, ex.toString());  
        }  
    }  
}  
  
@Override  
public IBinder onBind(Intent arg0) {  
    return null;  
}
```



메인 액티비티 코드 만들기

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Intent myIntent = new Intent(this, MyService.class );  
        startService(myIntent);  
    }  
}
```

The diagram illustrates the sequence of operations in the code. It consists of two blue circles with numbers and arrows pointing to specific lines of code. Circle 1, labeled '인텐트 객체 생성' (Intent object creation), points to the line 'Intent myIntent = new Intent(this, MyService.class);'. Circle 2, labeled '서비스 시작' (Service start), points to the line 'startService(myIntent);'.

매니페스트에 추가하기

```
...  
<application android:icon="@drawable/icon" android:label="@string/app_name">  
...  
    <service android:name="MyService" >  
        </service>  
  
</application>  
...
```



서비스 실행하여 로그 확인

[서비스를 통해 로그를 남긴 경우]

The screenshot shows the Android LogCat window with the following log entries:

Time	pid	tag	Message
09-19 09:40:27.754	D	ddm-heap	Got feature list request
09-19 09:40:28.355	I	9746 MyService	my service called #0
09-19 09:40:28.544	I	65 ActivityManager	Displayed activity org.androidtown
09-19 09:40:33.364	I	9746 MyService	my service called #1
09-19 09:40:33.754	D	243 dalvikvm	GC freed 6 objects / 176 bytes in :
09-19 09:40:38.367	I	9746 MyService	my service called #2
09-19 09:40:38.815	D	144 dalvikvm	GC freed 2298 objects / 134760 bytes
09-19 09:40:43.364	I	9746 MyService	my service called #3
09-19 09:40:48.372	I	9746 MyService	my service called #4
09-19 09:40:53.374	I	9746 MyService	my service called #5
09-19 09:40:58.377	I	9746 MyService	my service called #6
09-19 09:41:03.379	I	9746 MyService	
09-19 09:41:08.382	I	9746 MyService	
09-19 09:41:13.384	I	9746 MyService	
09-19 09:41:18.385	I	9746 MyService	
09-19 09:41:23.387	I	9746 MyService	

Filter: []

메인 액티비티

서비스

스레드 실행
run()

매니페스트 파일
(AndroidManifest.xml)

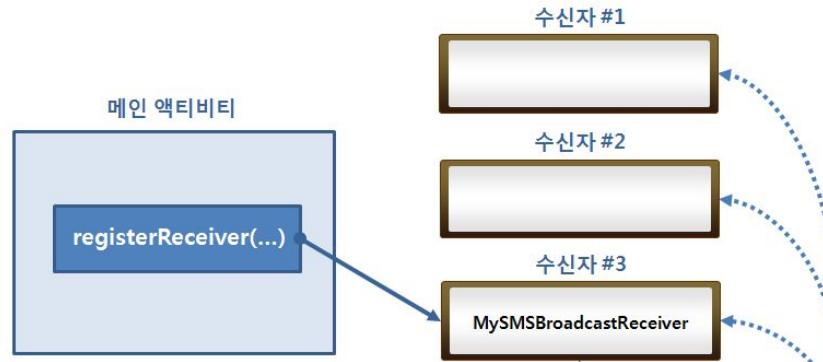
[서비스 이용 패턴]

The diagram illustrates the service utilization pattern. It shows a flow from the main activity's startService() method to the service's run() method, which is triggered by a manifest entry pointing to the service.

6.

브로드캐스트 수신자

브로드캐스트 수신자



- 애플리케이션이 글로벌 이벤트(global event)를 받아서 처리하려면 브로드캐스트 수신자로 등록
- 글로벌 이벤트란 “전화가 왔습니다.”, “문자 메시지가 도착했습니다.”와 같이 안드로이드 시스템 전체에 보내지는 이벤트
- 브로드캐스트 수신자는 인텐트필터를 포함하여, 매니페스트 파일에 등록함으로써 인텐트를 받을 준비를 함
- 수신자가 매니페스트 파일에 등록되었다면 따로 시작시키지 않아도 됨
- 애플리케이션은 컨텍스트 클래스의 `registerReceiver` 메소드를 이용하면 런타임 시에도 수신자를 등록할 수 있음
- 서비스처럼 브로드캐스트 수신자도 UI가 없음

브로드캐스트의 구분

• 인텐트와 브로드캐스트

- 인텐트를 이용해서 액티비티를 실행하면 포그라운드(foreground)로 실행되어 사용자에게 보여지지만
- 브로드캐스트를 이용해서 처리하면 백그라운드(background)로 동작하므로 사용자가 모름
- 인텐트를 받으면 `onReceive()` 메소드가 자동으로 호출됨

• 브로드캐스트의 구분

브로드캐스트는 크게 두 가지 클래스로 구분됨

- 일반 브로드캐스트 (`sendBroadcast()` 메소드로 호출)
비동기적으로 실행되며 모든 수신자는 순서없이 실행됨 (때로는 동시에 실행됨)
효율적이나, 한 수신자의 처리 결과를 다른 수신자가 이용할 수 없고 중간에 취소불가
- 순차 브로드캐스트 (`sendOrderedBroadcast()` 메소드로 호출)
한 번에 하나의 수신자에만 전달되므로 순서대로 실행됨. 중간에 취소하면 그 다음
수신자는 받지 못함. 수신자가 실행되는 순서는 인텐트필터의 속성으로 정할 수 있음
순서가 같으면 임의로 실행됨.

브로드캐스트 수신자 예제

브로드캐스트 수신자 예제

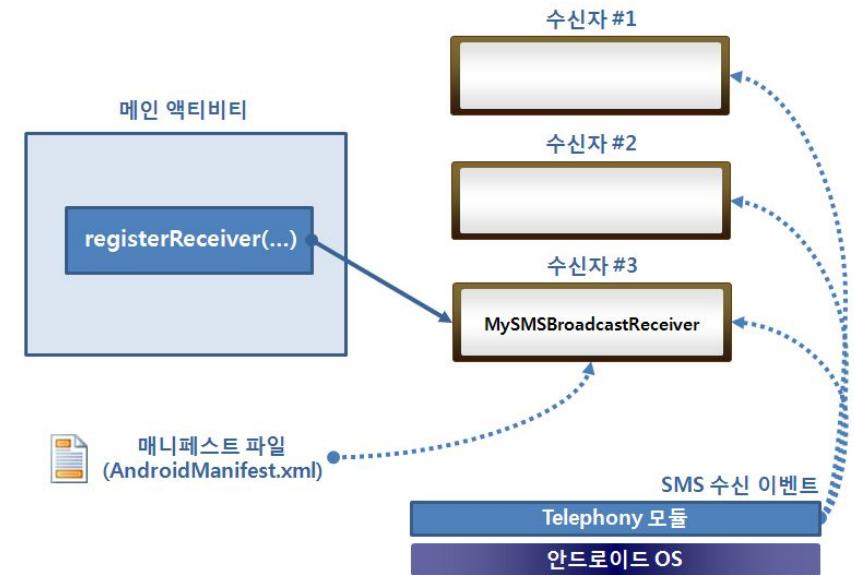
- 브로드캐스트 수신자로 SMS 수신 확인하기
- 브로드캐스트 수신자 정의

브로드캐스트 수신자 정의

- 일정 시간간격으로 메시지를 보여주는 서비스 클래스 정의

매니페스트에 추가

- 새로운 브로드캐스트 수신자를 매니페스트에 추가



브로드캐스트 수신자 클래스 정의

```
public class MySMSBroadcastReceiver extends BroadcastReceiver {  
    public void onReceive(Context context, Intent intent) {  
        Log.i("BroadcastReceiver", "onReceive");  
        if (intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {  
            Log.i("BroadcastReceiver", "SMS Event received!");  
  
            abortBroadcast();  
            Intent myIntent = new Intent(context, MainActivity.class);  
            myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
            context.startActivity(myIntent);  
        }  
    }  
}
```

1 브로드캐스트 메시지 수신 시 자동 호출됨

2 SMS_RECEIVED 액션인지를 확인

3 브로드캐스팅 취소

4 새로운 액티비티 띄우기

매니페스트에 추가하기

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

1

SMS 수신 권한 등록

...

```
<receiver android:name=".MySMSBroadcastReceiver">
```

```
    <intent-filter android:priority="10000">
```

```
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
```

```
    </intent-filter>
```

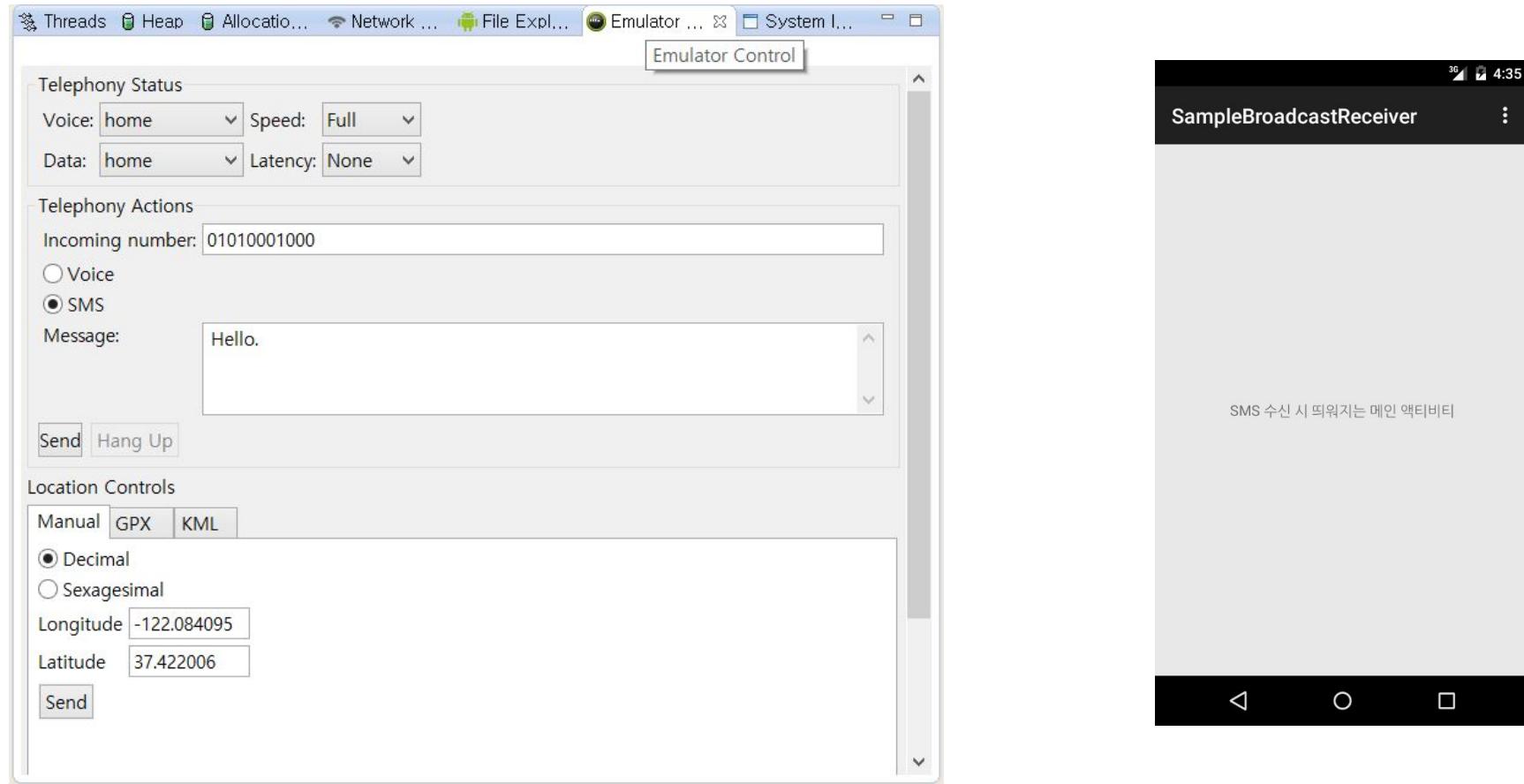
```
</receiver>
```

...

2

수신자 등록

브로드캐스트 수신자 실행 화면

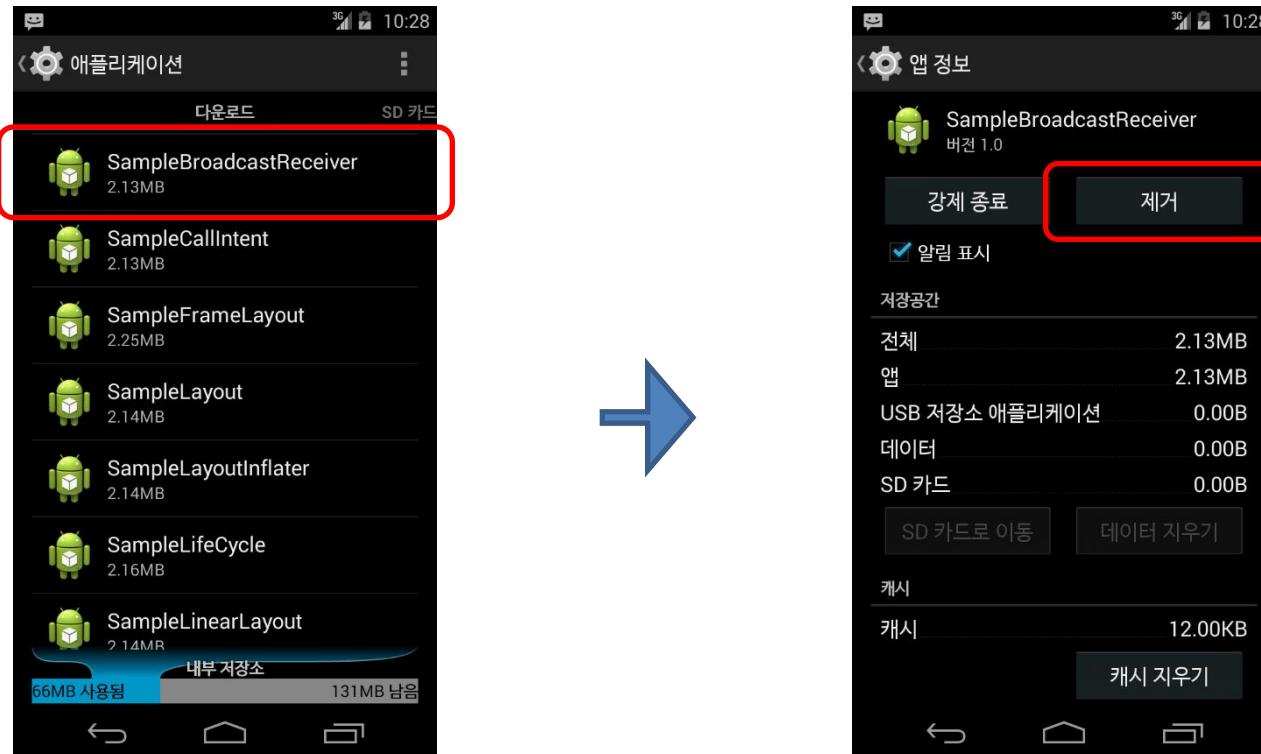


[DDMS에서 에뮬레이터로 SMS를 보내고 메인 액티비티가 뜬 화면]

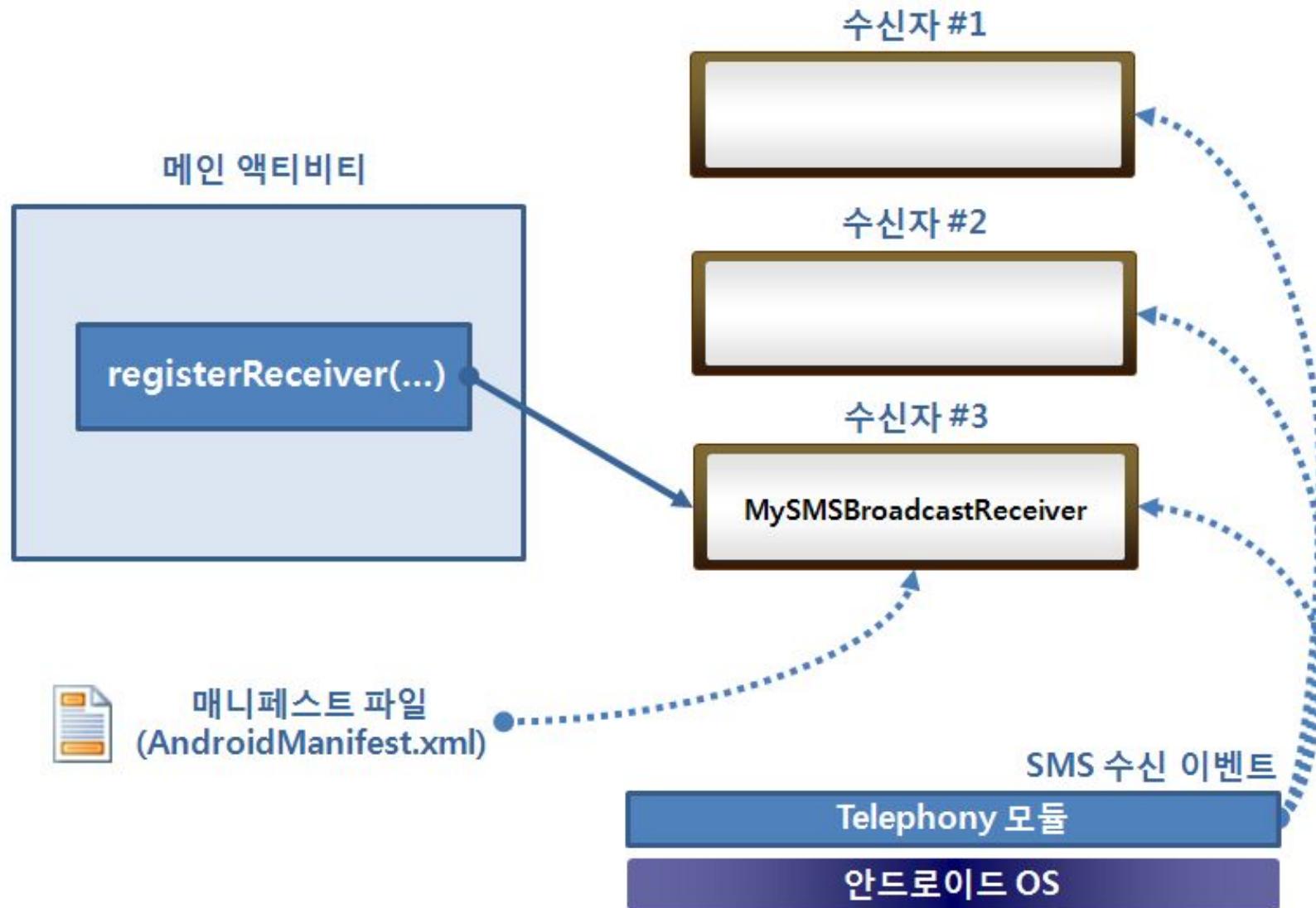
화면을 수정해도 이전 화면이 계속 보여요!

- **프로젝트를 복사하여 새로운 프로젝트를 만든 경우**

- SMS를 수신하는 동일한 애플리케이션이 설치되어 있다면 SMS 수신 시 그 화면이 보여질 수 있음
- 따라서, 이전에 설치한 애플리케이션을 삭제한 후 새로운 애플리케이션을 실행해야 함
- 단말의 설정에서 앱을 삭제할 수 있음



브로드캐스트 수신자 사용 패턴



7.

앱을 실행했을 때 권한 부여

일반 권한과 위험 권한 (마시멜로 API23부터)

- 위험 권한은 실행 시 권한 부여



대표적인 위험 권한들



LOCATION (위치)

- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION

CAMERA

- CAMERA

MICROPHONE

- RECORD_AUDIO

CONTACTS

- READ_CONTACTS

- WRITE_CONTACTS

- GET_ACCOUNTS

PHONE

- READ_PHONE_STATE

- CALL_PHONE

- READ_CALL_LOG

- WRITE_CALL_LOG

- ADD_VOICEMAIL

- USE_SIP

- PROCESS_OUTGOING_CALLS

SMS

- SEND_SMS

- RECEIVE_SMS

- READ_SMS

- RECEIVE_WAP_PUSH

- RECEIVE_MMS

CALENDAR

- READ_CALENDAR

- WRITE_CALENDAR

SENSORS

- BODY_SENSORS

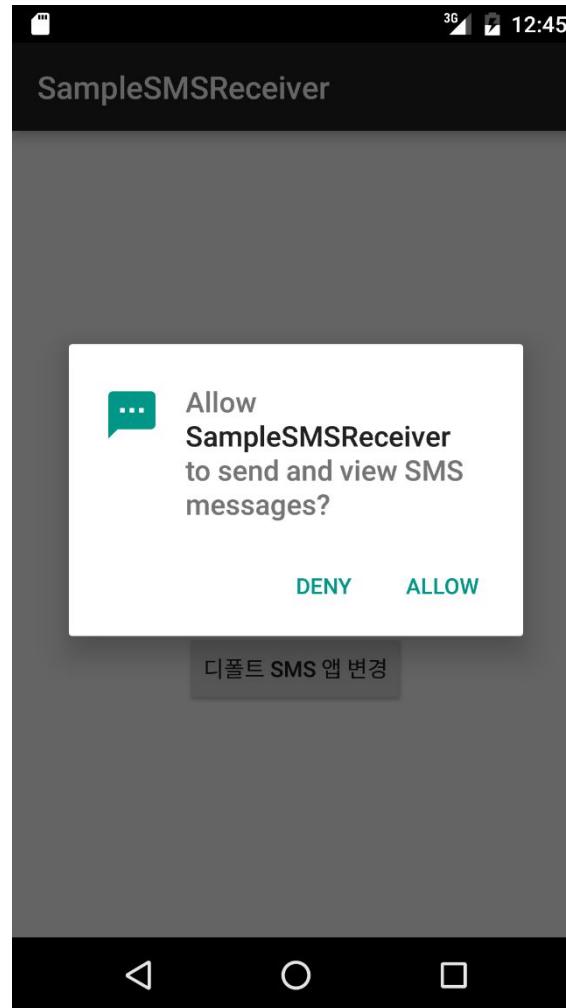
STORAGE

- READ_EXTERNAL_STORAGE

- WRITE_EXTERNAL_STORAGE

실행 시 권한 부여

- 실행 시 권한 부여를 묻는 대화상자 표시



권한 요청 코드

```
public class MainActivity extends AppCompatActivity {

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

        int permissionCheck = ContextCompat.checkSelfPermission(this,
                Manifest.permission.RECEIVE_SMS);
        if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "SMS 수신 권한 있음.", Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(this, "SMS 수신 권한 없음.", Toast.LENGTH_LONG).show();
            if (ActivityCompat.shouldShowRequestPermissionRationale(
                    this, Manifest.permission.RECEIVE_SMS)) {
                Toast.makeText(this, "SMS 권한 설명 필요함.",
                        Toast.LENGTH_LONG).show();
            } else {
                ActivityCompat.requestPermissions(this,
                        new String[] {Manifest.permission.RECEIVE_SMS},
                        1);
            }
        }
    }
}
```

①

②

권한 요청 결과 확인 코드

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) { 
        case 1: {
            if (grantResults.length > 0 &&
                grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "SMS 권한을 사용자가 승인함.", Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(this, "SMS 권한 거부됨.", Toast.LENGTH_LONG).show();
            }
        }
    }
}
```



8.

리소스와 매니페스트

매니페스트의 태그 항목

[Reference]

```
<action><permission>
<activity><permission-group>
<activity-alias>
<application><provider>
<category><receiver>
<data><service>
<grant_uri_permission><uses_configuration>
<instrumentation><uses-library>
<intent-filter><uses-permission>
<manifest><uses-sdk>
<meta-data>
```

매니페스트 파일의 역할

- 애플리케이션의 자바 패키지 이름 지정
- 애플리케이션 구성요소에 대한 정보 등록(액티비티, 서비스, 브로드캐스트 수신자, 내용 제공자)
- 각 구성요소를 구현하는 클래스 이름 지정
- 애플리케이션이 가져야 하는 권한에 대한 정보 등록
- 다른 애플리케이션이 접근하기 위해 필요한 권한에 대한 정보 등록
- 애플리케이션 개발 과정에서 프로파일링을 위해 필요한 instrumentation 클래스 등록
- 애플리케이션에 필요한 안드로이드 API의 레벨 정보 등록
- 애플리케이션에서 사용하는 라이브러리 리스트

[Code]

[매니페스트 파일의 기본 구조]

```
<manifest ... >
<application ... >
...
<service android:name="org.androidtown.service.MyService" ... >
...
</service>
...
</application>
</manifest>
```

메인 액티비티 정의

[Code]

```
<activity android:name="org.androidtown.basicMainActivity"  
        android:label="@string/app_name">  
  
    <intent-filter>  
  
        <action android:name="android.intent.action.MAIN" />  
  
        <category android:name="android.intent.category.LAUNCHER" />  
  
    </intent-filter>  
  
</activity>
```

리소스 사용

- 리소스를 자바 코드와 분리하는 이유는 이해하기 쉽고 유지관리가 용이하기 때문임
- 프로젝트를 처음에 만들면 **[/res]** 폴더와 **[/assets]** 폴더가 따로 분리되어 있는데
두 가지 모두 리소스라고 할 수 있으며 대부분은 **[/res]** 폴더 밑에서 관리됨

- 애셋(Asset)은 동영상이나 웹페이지와 같이 용량이 큰 데이터를 의미함
- 리소스는 빌드되어 설치파일에 추가되지만 애셋은 빌드되지 않음

스타일과 테마

- 스타일과 테마는 여러 가지 속성들을 한꺼번에 모아서 정의한 것
- 대표적인 예로는 대화상자를 들 수 있음

[Code]

```
<style name="Alert" parent="android:Theme.Dialog">  
    <item name="android:windowBackground">@drawable/alertBackground</item>  
</style>
```

9.

토스트와 대화상자

토스트와 대화상자

- 토스트

- 간단한 메시지를 잠깐 보여주었다가 없어지는 뷰로 애플리케이션 위에 떠 있는 뷰라 할 수 있음

[Code]

```
Toast.makeText(Context context, String message, int duration)
```

[Code]

```
public void setGravity(int gravity, int xOffset, int yOffset)
```

```
public void setMargin(float horizontalMargin, float verticalMargin)
```

토스트 만들기 예제

토스트 만들기 예제

- 토스트의 색상이나 모양을 직접 구성
- 새로운 레이아웃 정의

메인 액티비티
XML 레이아웃 정의

-메인 액티비티의 레이아웃 정의

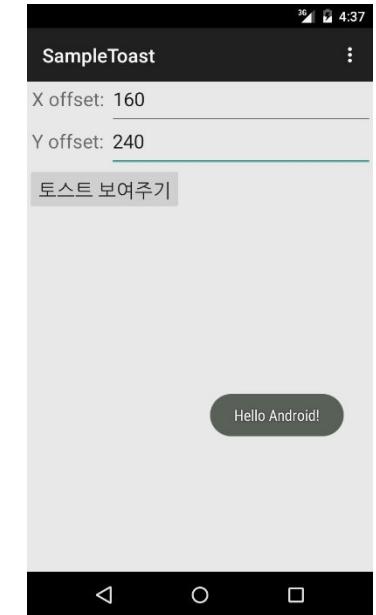
메인 액티비티 코드 작성

-메인 액티비티에서 위치 설정

토스트를 위한
XML 레이아웃 정의

메인 액티비티 코드 작성

-토스트의 모양을 XML 레이아웃으로 메인 액티비티에서 모양 설정
정의



메인 액티비티 코드 만들기

```
Toast toastView = Toast.makeText(getApplicationContext(),  
    "Hello Android!",  
    Toast.LENGTH_LONG);  
  
int xOffset = Integer.valueOf(edit01.getText().toString());  
int yOffset = Integer.valueOf(edit02.getText().toString());  
toastView.setGravity(Gravity.CENTER, xOffset, yOffset);  
toastView.show();
```

2 토스트 객체 생성

3 x offset 값 확인

4 y offset 값 확인

5 토스트가 보일 위치 지정

6 토스트 보이기

토스트 모양 바꾸기 – 메인 액티비티 코드 만들기

```
LayoutInflator inflater = getLayoutInflater();
View layout = inflater.inflate(
    R.layout.toastborder,
    (ViewGroup) findViewById(R.id.toast_layout_root));
TextView text = (TextView) layout.findViewById(R.id.text);
Toast toast = new Toast(getApplicationContext());
text.setText("Hello My Android!");
toast.setGravity(Gravity.CENTER, 0, 0);
toast.setDuration(Toast.LENGTH_SHORT);
toast.setView(layout);
toast.show();
```

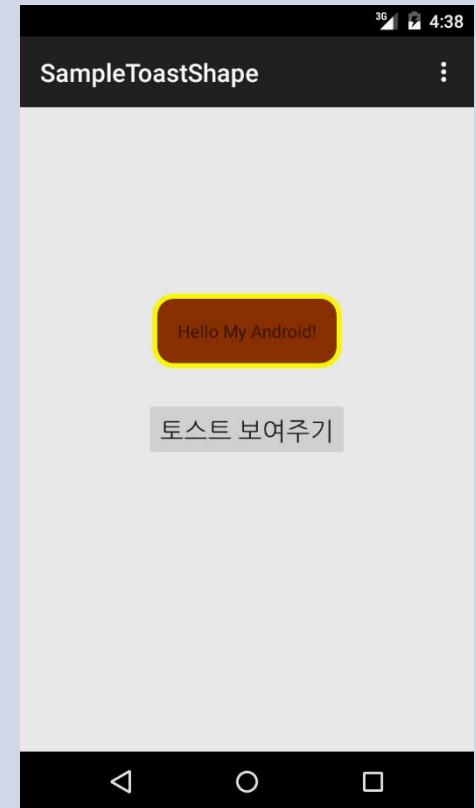
1 레이아웃 인플레이터 객체 참조
2 토스트를 위한 레이아웃 인플레이션
3 토스트 객체 생성
4 토스트가 보이는 뷰 설정

토스트 모양 바꾸기 – 토스트의 XML 레이아웃

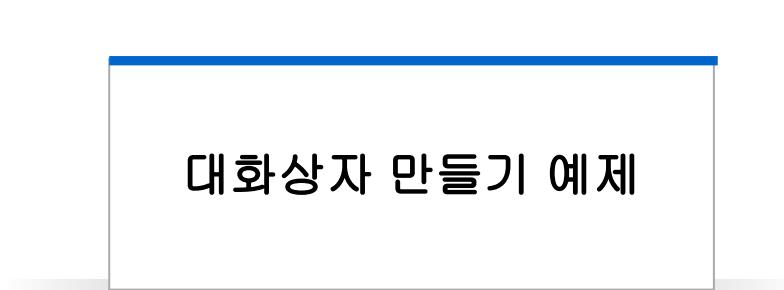
```
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/toast_layout_root"  
    android:orientation="horizontal"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:padding="10dp"  
    >  
    <TextView  
        android:id="@+id/text"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:padding="20dp"  
        android:background="@drawable/toast"  
    />  
    </LinearLayout>
```

Shape 객체를 위한 XML 정의

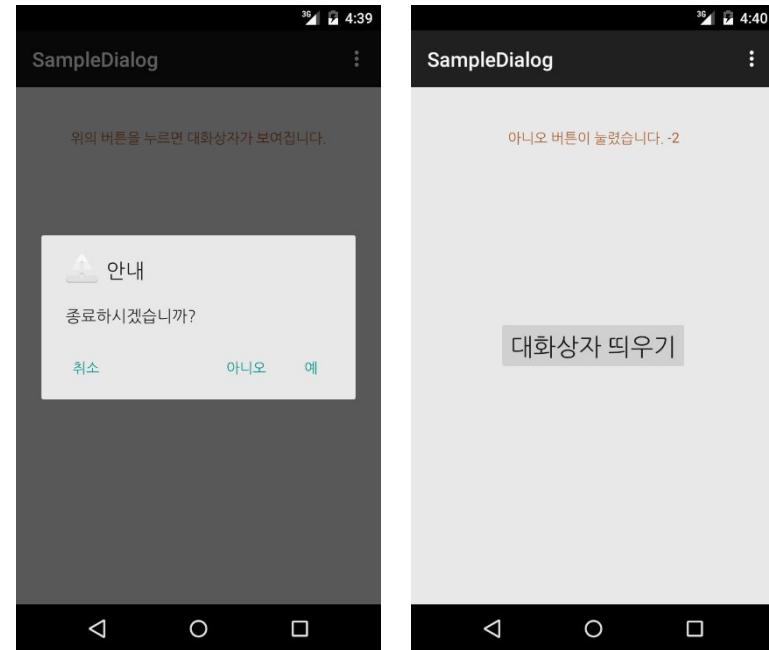
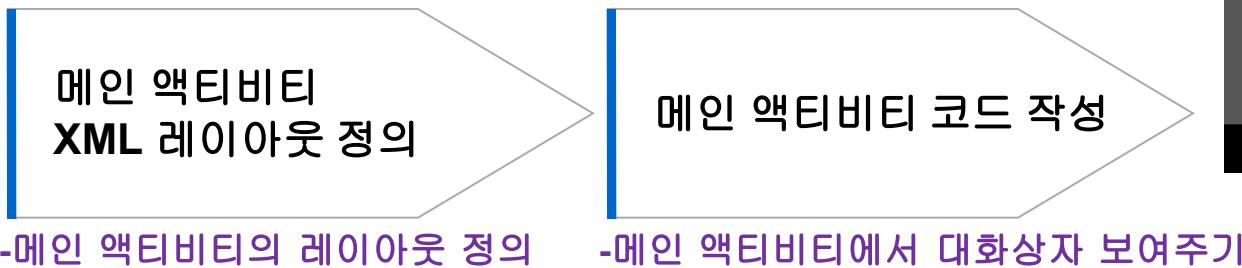
```
<?xml version="1.0" encoding="UTF-8" ?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle"
    >
    <stroke
        android:width="4dp"
        android:color="#ffff00"
    />
    <solid
        android:color="#ff883300"
    />
    <padding
        android:left="20dp"
        android:top="20dp"
        android:right="20dp"
        android:bottom="20dp"
    />
    <corners
        android:radius="15dp"
    />
</shape>
```



대화상자 만들기 예제



- 대화상자 보여주기
 - 이벤트 처리



메인 액티비티 코드 만들기

```
AlertDialog dialog = createDialogBox();  
dialog.show();
```

1 createDialogBox() 메소드 호출하여 대화상자 객체 생성

2 대화상자 보여주기

private AlertDialog createDialogBox()

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
```

```
builder.setTitle("안내");  
builder.setMessage("종료하시겠습니까?");  
builder.setIcon(R.drawable.alert_dialog_icon);
```

3 대화상자의 타이틀, 메시지, 아이콘 설정

Continued..

메인 액티비티 코드 만들기 (계속)

```
builder.setPositiveButton("예", new DialogInterface.OnClickListener()
    public void onClick(DialogInterface dialog, int whichButton)
        msg = "예 버튼이 눌렸습니다. " + Integer.toString(whichButton);
        txtMsg.setText(msg);

    );

builder.setNeutralButton("취소",new DialogInterface.OnClickListener()
    public void onClick(DialogInterface dialog, int whichButton)
        msg = "취소 버튼이 눌렸습니다. " + Integer.toString(whichButton);
        txtMsg.setText(msg);

    );

builder.setNegativeButton("아니오", new DialogInterface.OnClickListener()
    public void onClick(DialogInterface dialog, int whichButton)
        msg = "아니오 버튼이 눌렸습니다. " + Integer.toString(whichButton);
        txtMsg.setText(msg);

    );

AlertDialog dialog = builder.create();

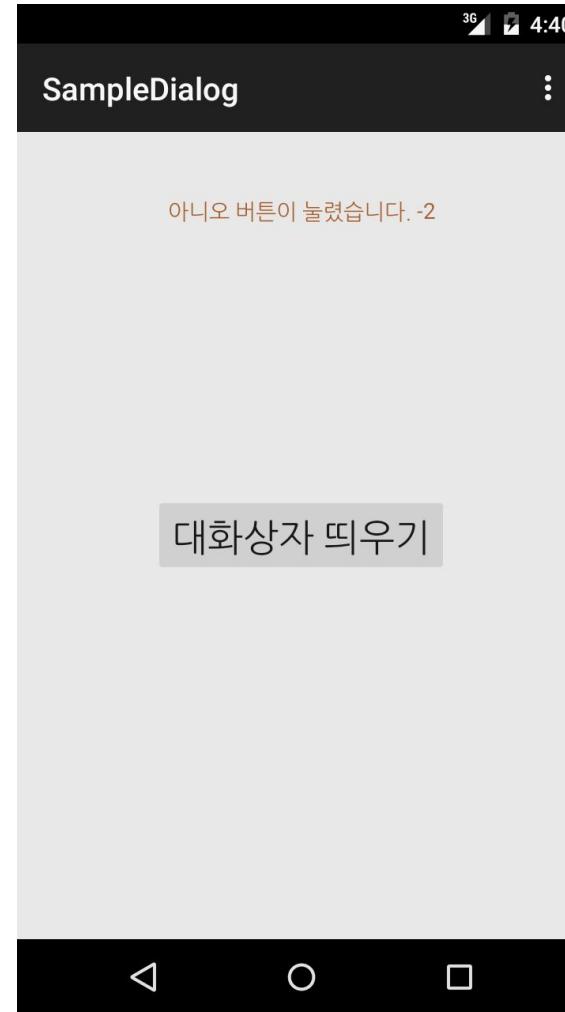
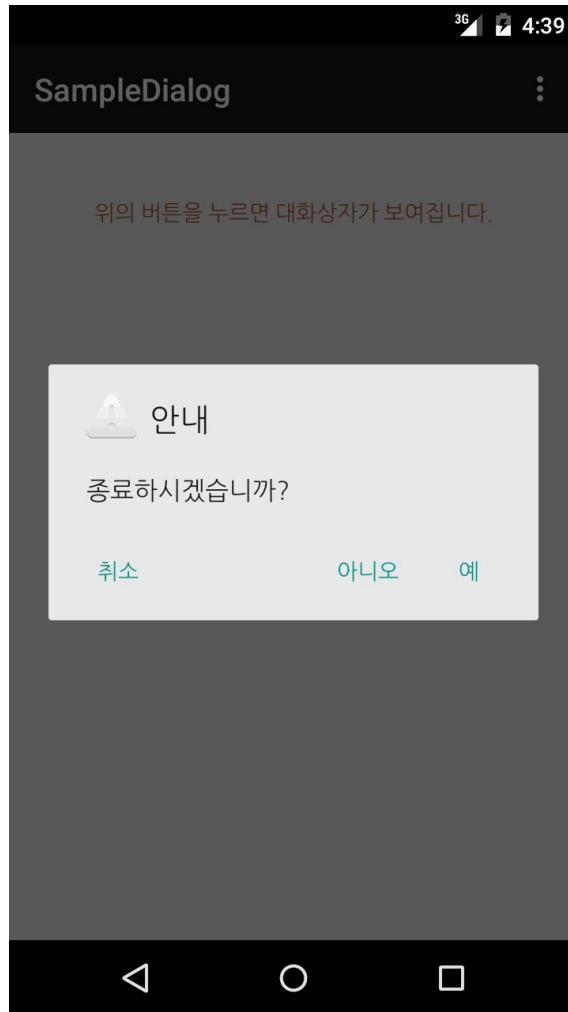
return dialog;
}
```

4 “예” 버튼 기능 설정

5 “취소” 버튼 기능 설정

6 “아니오” 버튼 기능 설정

대화상자 만들기 실행 화면



10.

프래그먼트

프래그먼트란?

- **프래그먼트 (Fragment)**

- 화면의 일정 영역을 독립적으로 처리하기 위해 만들어진 특별한 화면 구성 요소
- 태블릿의 대화면에서 화면 분할이 필요하게 되면서 만들어짐

- **프래그먼트의 기본 목적**

- 하나의 화면이 XML 레이아웃과 자바 소스로 구성된다는 점에 착안하여 하나의 프래그먼트가 XML 레이아웃과 자바 소스로 구성되도록 하고 독립적으로 관리되도록 하기 위함

- **기본 구성 방식**

- 액티비티가 동작하는 방식을 본떠 만들었음

- **사용되는 개념**

- 프래그먼트 매니저 : 프래그먼트를 관리하는 객체
- 트랜잭션 : 프래그먼트의 처리를 위해 만든 단위

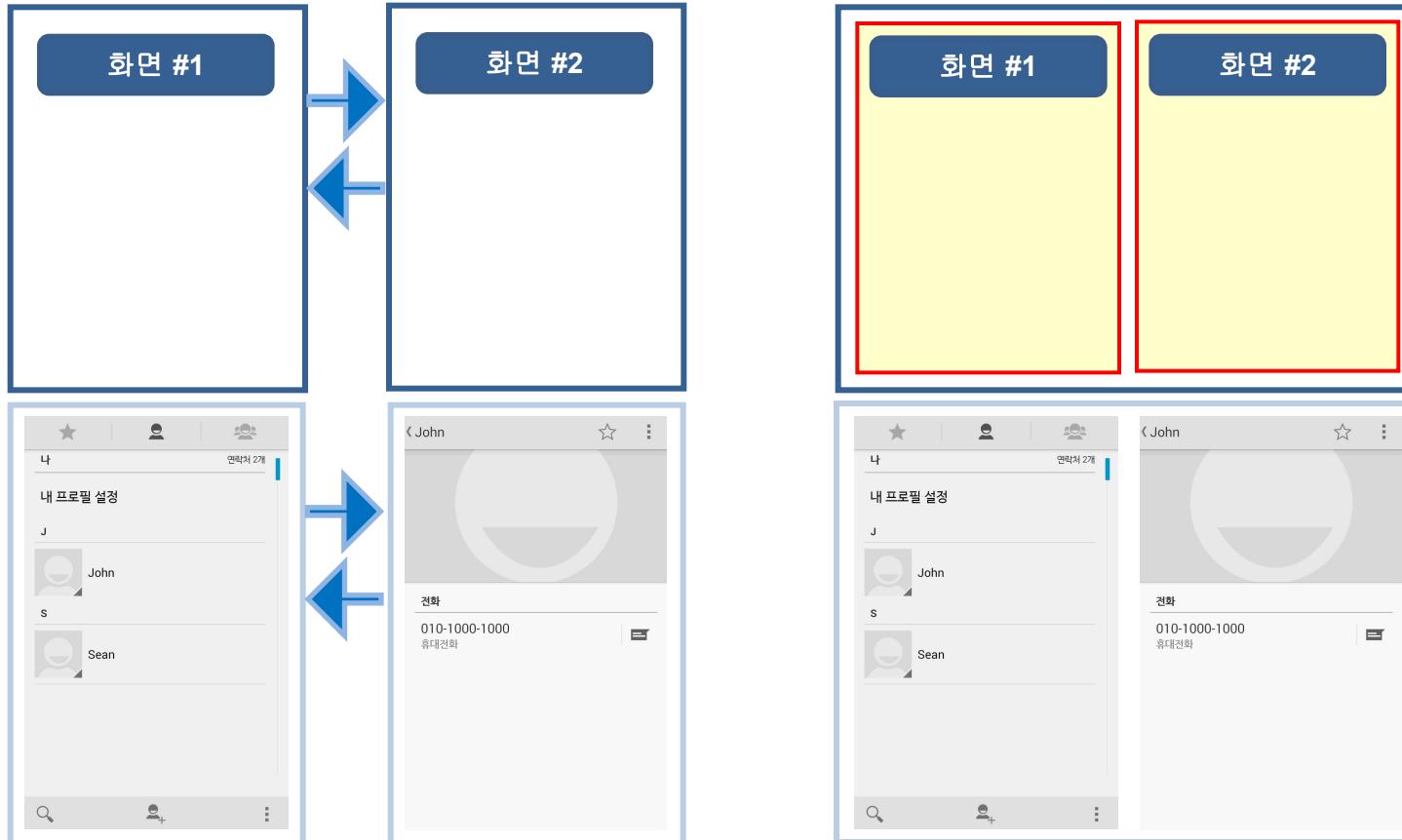
화면을 전환하는 경우와 화면을 분할하는 경우

• 화면 전환과 화면 분할

(1) 두 개의 화면을 액티비티로 만들고 액티비티 간 전환

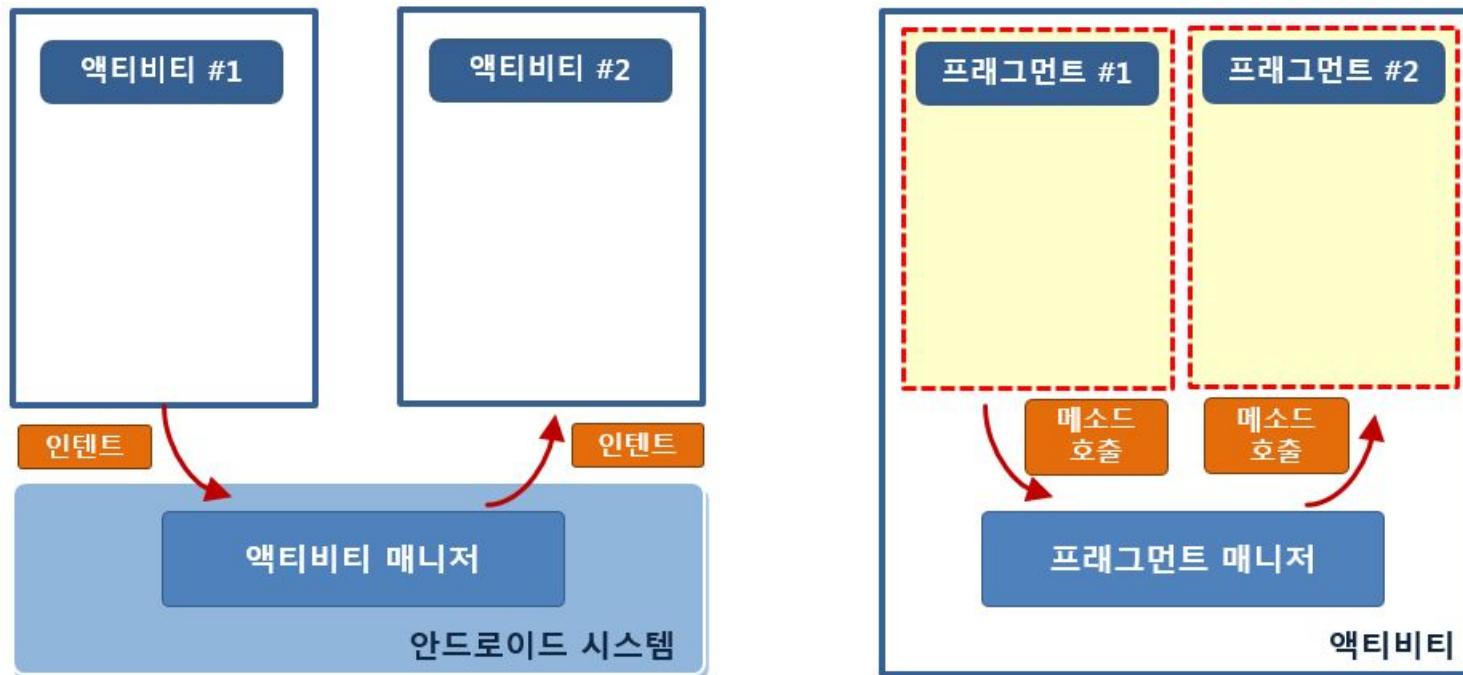
(2) 하나의 액티비티 위에 프래그먼트를 두고 프래그먼트 간 전환

✓ 프래그먼트로 만들어 두면 스마트폰에서는 프래그먼트 간 화면 전환, 태블릿에서는 두 개의 프래그먼트를 하나의 액티비티 위에서 동시에 보여주는 화면 분할이 가능함



액티비티와 프래그먼트의 동작 방식 비교

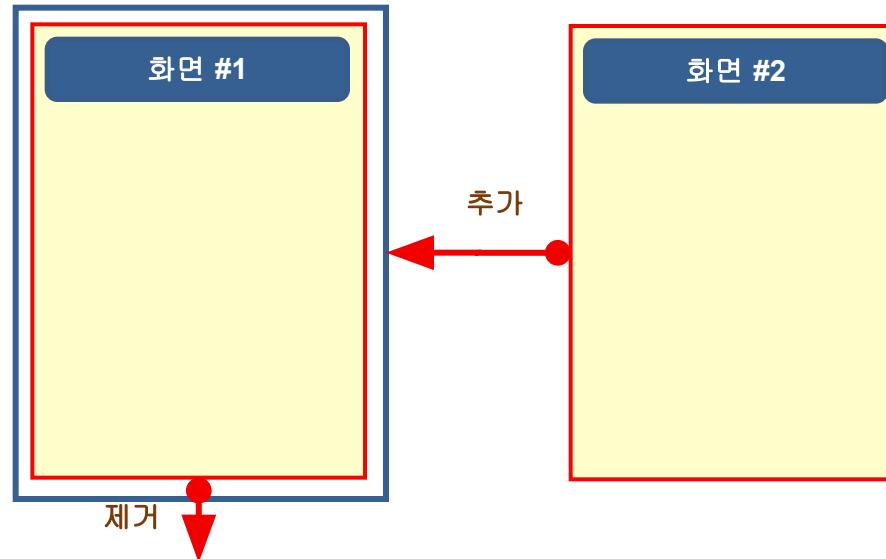
- 액티비티를 시스템의 액티비티 매니저에서 관리하듯이 프래그먼트를 액티비티의 프래그먼트 매니저에서 관리함
 - 프래그먼트 입장에서는 액티비티가 시스템 역할을 하므로 프래그먼트는 항상 액티비티 위에 올라가 있어야 함



하나의 액티비티 안에서 프래그먼트 전환

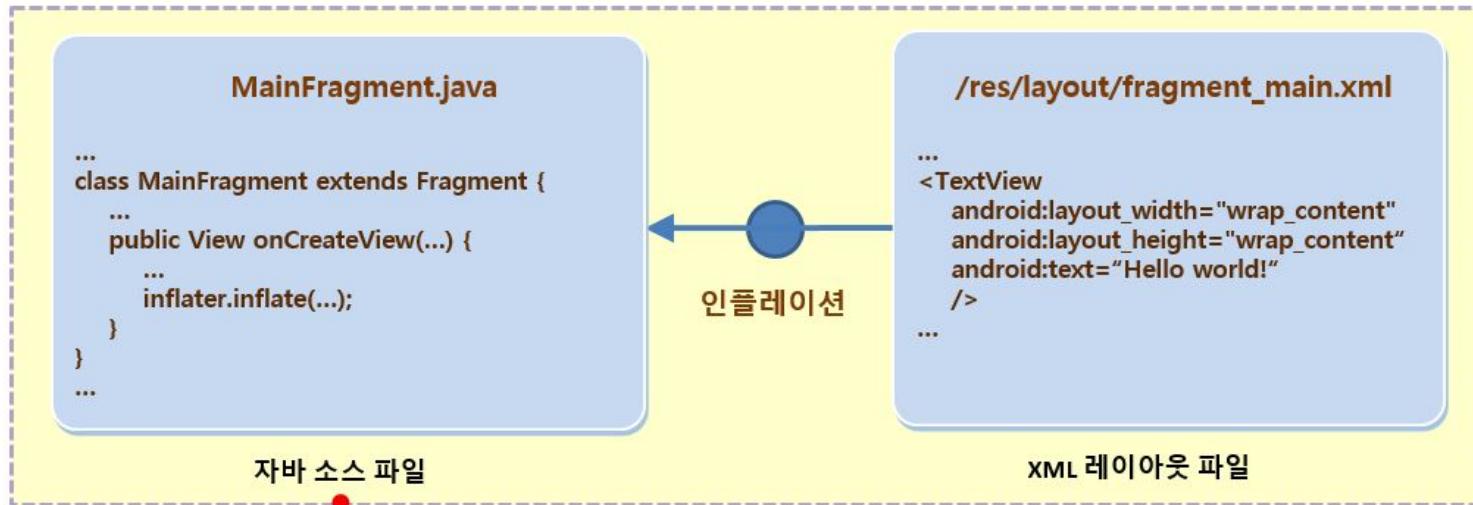
- **프래그먼트 전환**

- 프래그먼트 매니저와 트랜잭션을 이용해 추가(add)나 교체(replace) 가능
- 싱글 프래그먼트라고 부르며 액티비티 전환 없이 화면 전체가 전환되는 효과를 낼 수 있음



프로그먼트를 위한 레이아웃 인플레이션

하나의 프로그먼트



```
<fragment  
    android:id="@+id/fragment"  
    android:name="org.androidtown.fragment.MainFragment"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

Fragment 클래스

public final Activity getActivity ()

이 프래그먼트를 포함하는 액티비티를 리턴함.

public final FragmentManager getFragmentManager ()

이 프래그먼트를 포함하는 액티비티에서 프래그먼트 객체들과 의사소통하는
프래그먼트 매니저를 리턴함.

public final Fragment getParentFragment ()

이 프래그먼트를 포함하는 부모가 프래그먼트일 경우 리턴함. 액티비티이면 null을
리턴함.

public final int getId ()

이 프래그먼트의 ID를 리턴함.

FragmentManager 클래스

public abstract FragmentTransaction beginTransaction ()

프래그먼트를 변경하기 위한 트랜잭션을 시작함.

public abstract Fragment findFragmentById (int id)

ID를 이용해 프래그먼트 객체를 찾음.

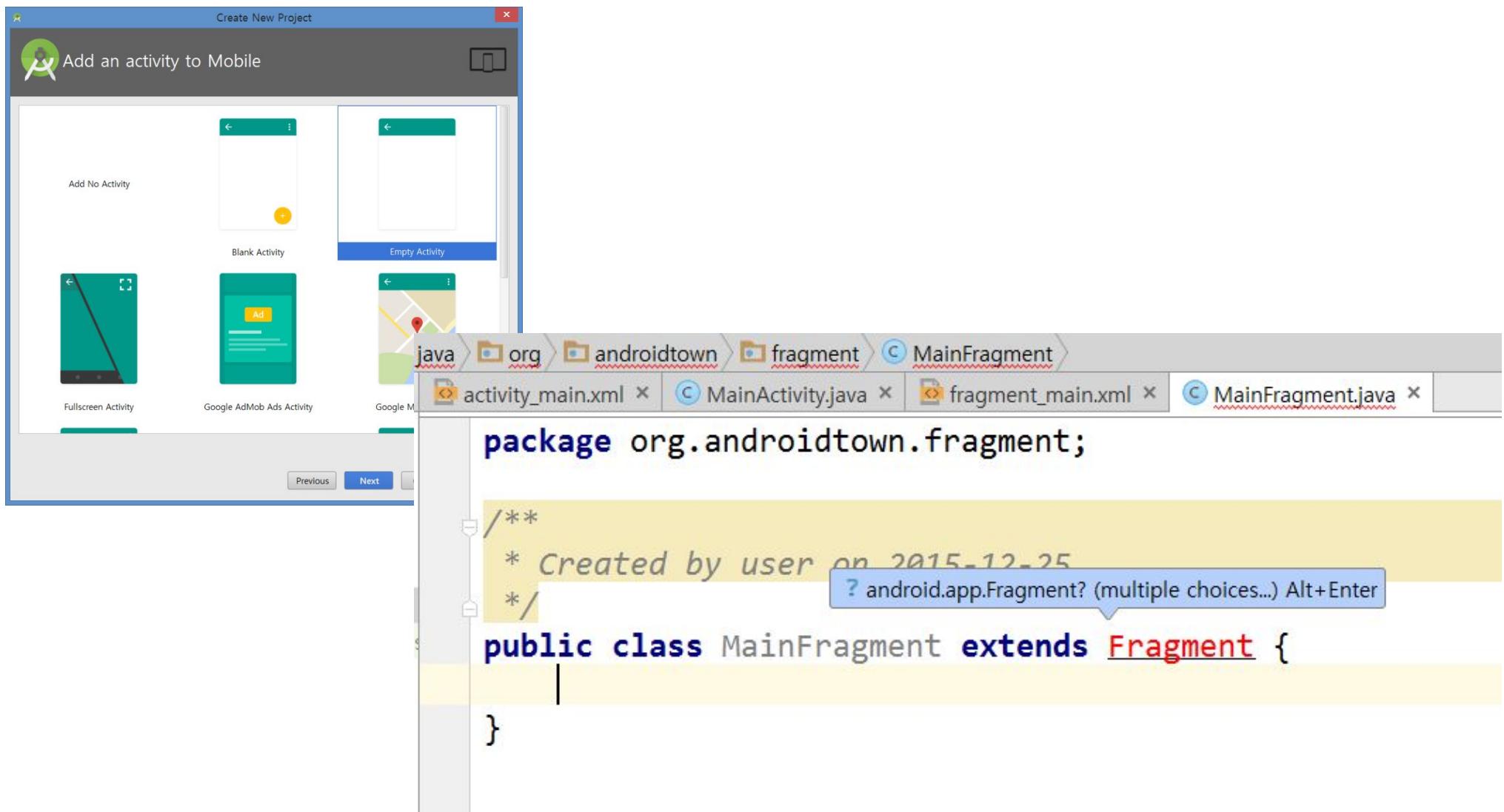
public abstract Fragment public abstract Fragment findFragmentByTag (String tag)

태그 정보를 이용해 프래그먼트 객체를 찾음.

public abstract boolean executePendingTransactions ()

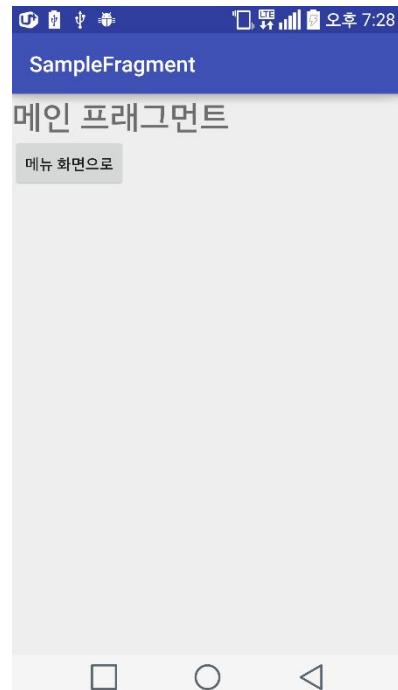
트랜잭션은 commit() 메소드를 호출하면 실행되지만 비동기(asynchronous) 방식으로 실행되므로 즉시 실행하고 싶다면 이 메소드를 추가로 호출해야 함.

프래그먼트 프로젝트 만들기

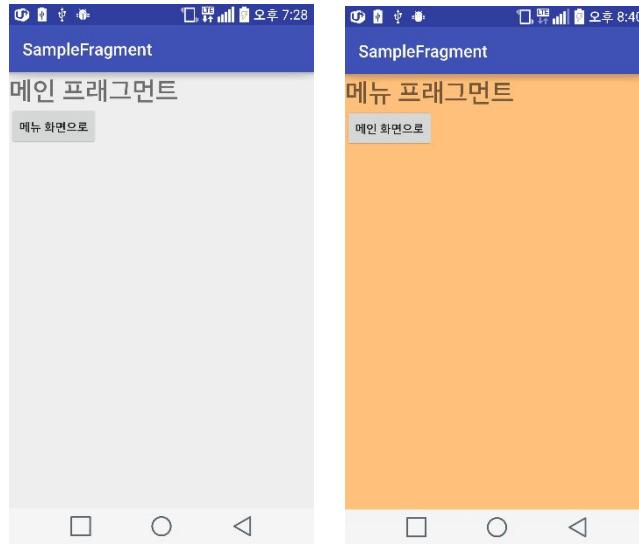


프래그먼트를 만들어 사용하는 과정

- (1) 프래그먼트를 위한 XML 레이아웃 만들기
- (2) 프래그먼트 클래스 만들기 (클래스 정의)
- (3) `onCreateView()` 메소드 안에서 인플레이션하기
- (4) 메인 액티비티를 위한 XML 레이아웃에 추가하거나 프래그먼트 매니저를 이용해 코드에서 추가하기



프래그먼트의 교체



```
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                        @Nullable Bundle savedInstanceState) {
    ViewGroup rootView = (ViewGroup) inflater.inflate(R.layout.fragment_main,
                                                    container, false);

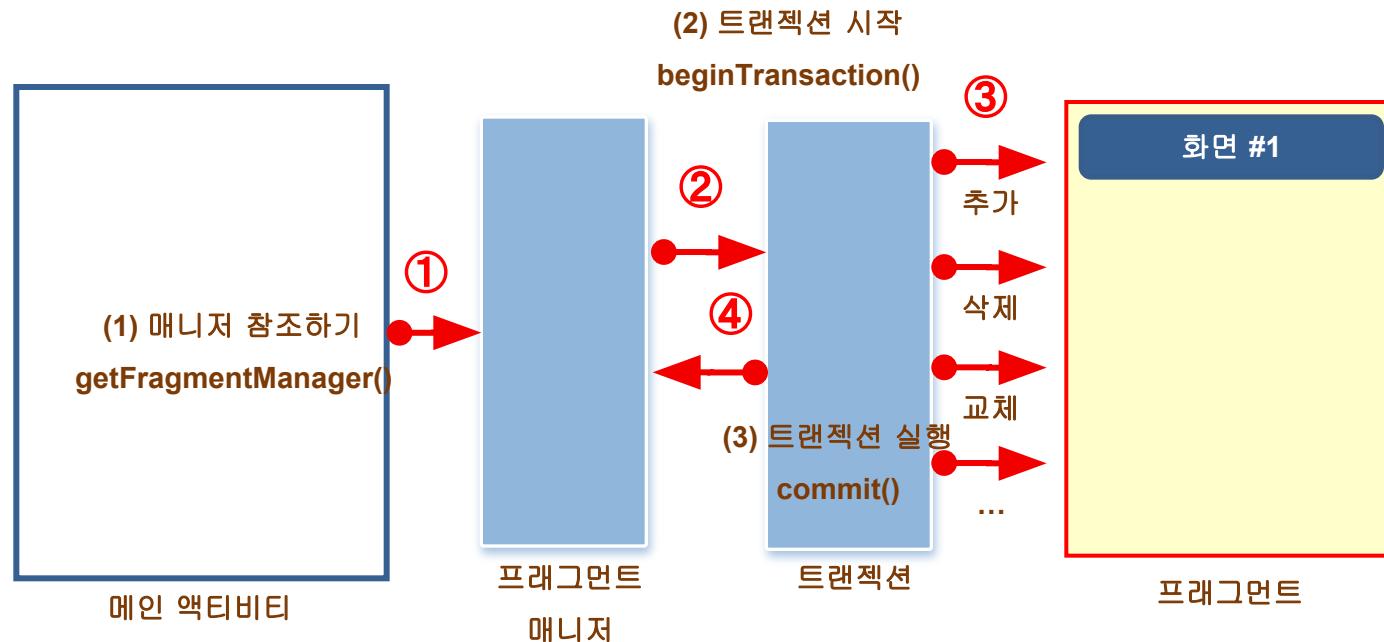
    Button button = (Button) rootView.findViewById(R.id.button);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            MainActivity activity = (MainActivity) getActivity();
            activity.onFragmentChanged(0);
        }
    });
}

return rootView;
}
```

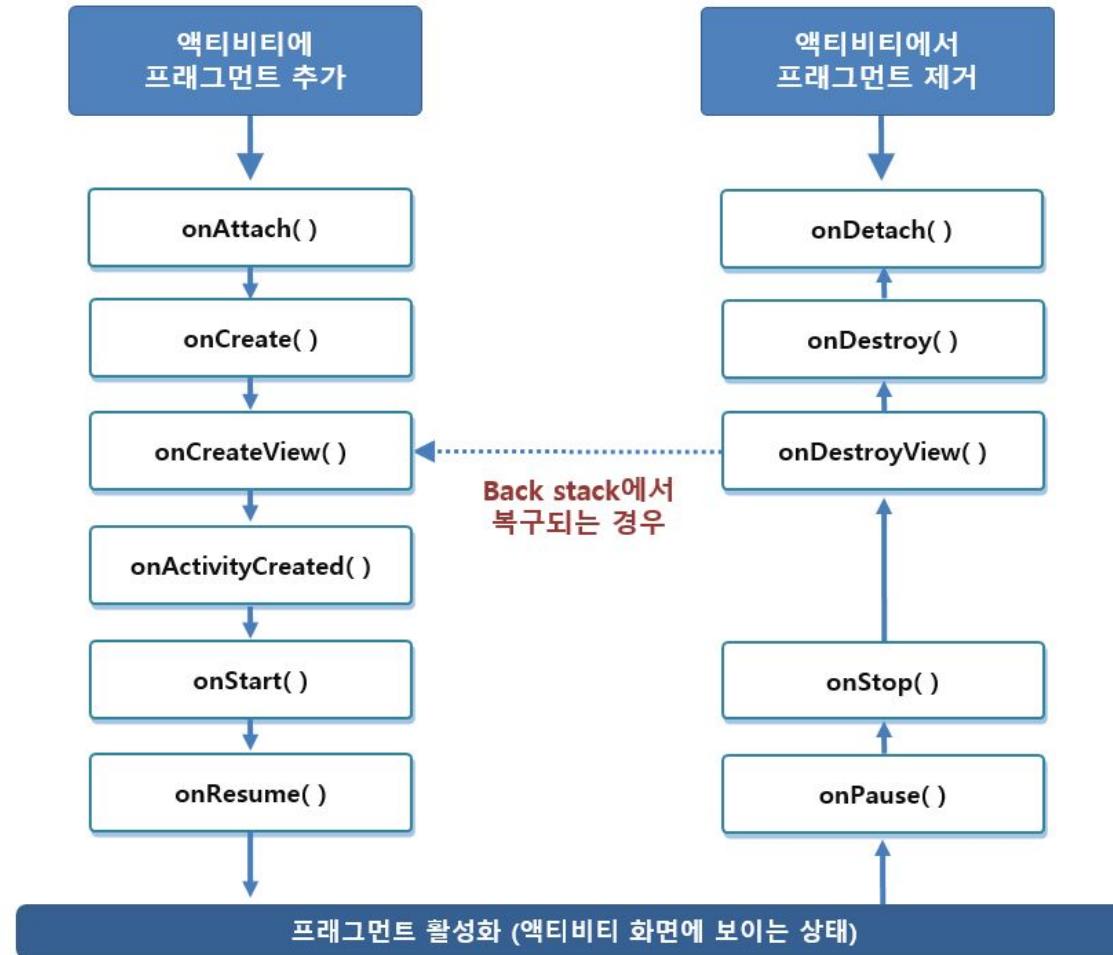
액티비티와 프래그먼트 간 의사소통

- 프래그먼트 처리 순서

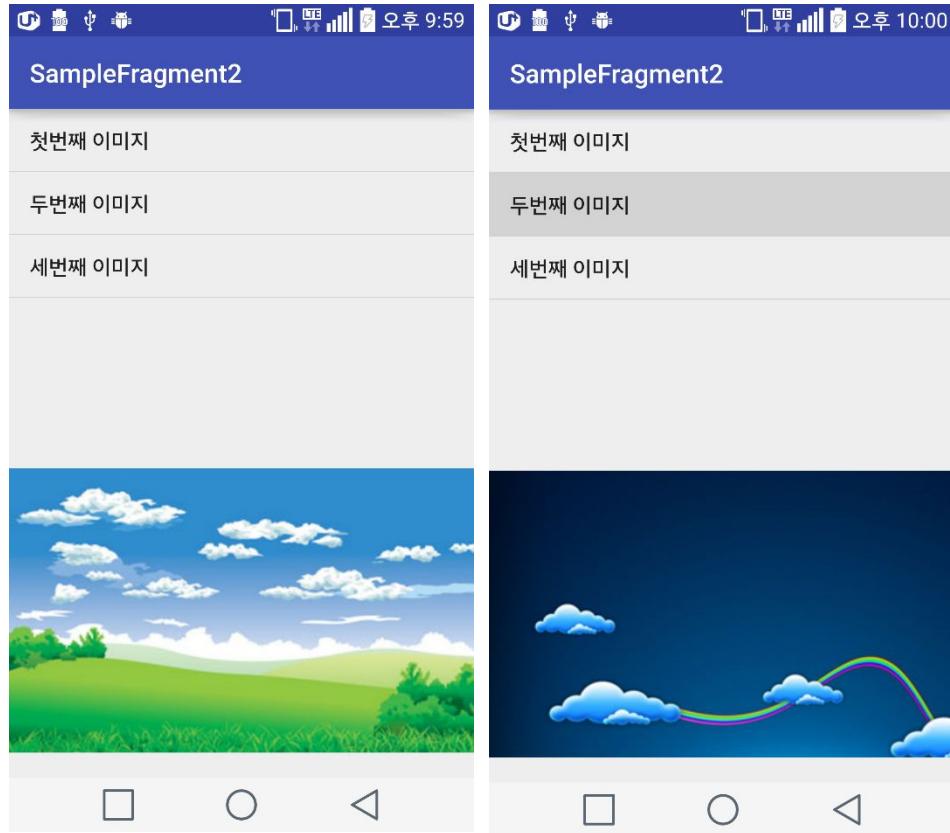
- (1) 프래그먼트 매니저 객체 참조
- (2) 트랜잭션 시작
- (3) 프래그먼트의 추가, 삭제 또는 교체
- (4) 트랜잭션 커밋 (commit)



프래그먼트의 수명주기



한 화면에 두 개의 프래그먼트 넣기



```
<fragment  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:layout_weight="1"  
    android:name="org.androidtown.fragment.ListFragment"  
    android:id="@+id/listFragment"  
/>  
  
<fragment  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:layout_weight="1"  
    android:name="org.androidtown.fragment.ViewerFragment"  
    android:id="@+id/viewerFragment"  
/>
```

액티비티의 코드에서 프래그먼트 전환

```
FragmentManager manager = getSupportFragmentManager();
listFragment = (ListFragment) manager.findFragmentById(R.id.listFragment);
viewerFragment = (ViewerFragment) manager.findFragmentById(R.id.viewerFragment);
```

```
@Override
public void onImageSelected(int position) {
    viewerFragment.setImage(images[position]);
}
```

[References]

- 기본 서적

2016, 정재곤, “Do it! 안드로이드 앱 프로그래밍(개정3판)”, 이지스퍼블리싱(주)

- Android Website

<http://www.android.com/>

- Google Developer's Conference

<http://code.google.com/events/io/>

- Android SDK Documentation