Do It! 안드로이드 앱 프로그래밍

개정6판

안드로이드 스튜디오 3.3 이상 기준

Mar. 2019

저자 : 정재곤

Do It! 안드로이드 앱 프로그래밍

둘째 마당 - Chapter 02

애플리케이션 구성하기

이지스퍼블리싱(주) 제공 강의 교안

저자 : 정재곤

이번 장에서는 무엇을 다룰까요?

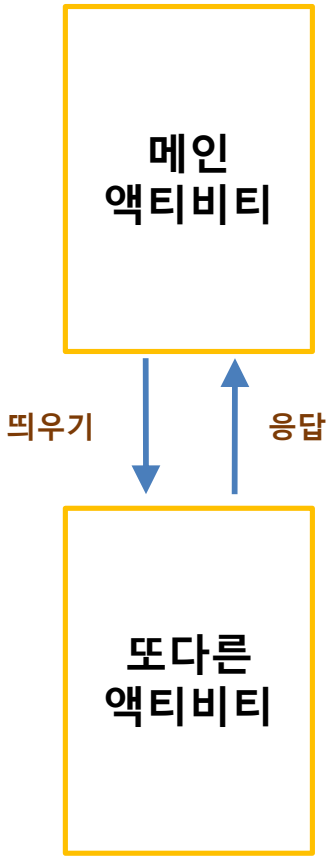
여러 화면들이 있는 제대로 된 애플리케이션을 만들고 싶어요.

- XML로 만든 레이아웃이 어떻게 화면에 보여지는 것일까요?
- 여러 화면들을 만들고 나면 어떻게 화면을 전환시킬까요?
- 화면이 바뀔 때 데이터는 어떻게 전달할까요?
- 화면이 없는 기능은 어떻게 만드는 것이 좋을까요?
- SMS 수신할 때 애플리케이션에서 알 수 있는 방법이 있나요?
- 리소스와 매니페스트에 대해 좀더 알아볼까요?
- 간단하게 사용할 수 있는 대화상자로는 어떤 것이 있을까요?

이번 장에서는 무엇을 다룰까요?

레이아웃 인플레이션

화면 전환



애플리케이션 구성요소

강의 주제 및 목차

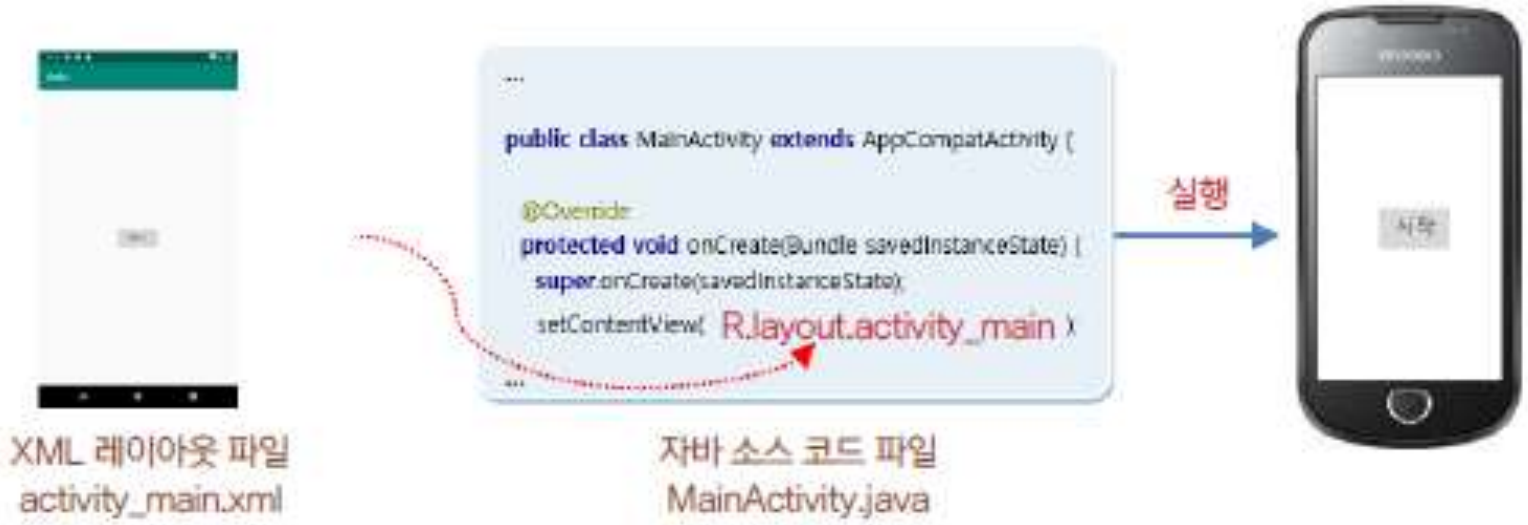
강의 주제

안드로이드 애플리케이션 구성요소에 대한 이해

1. 레이아웃 인플레이션

XML 레이아웃 파일과 자바 소스 파일의 매칭

- setContentView 메소드에서 XML 레이아웃 파일 매칭

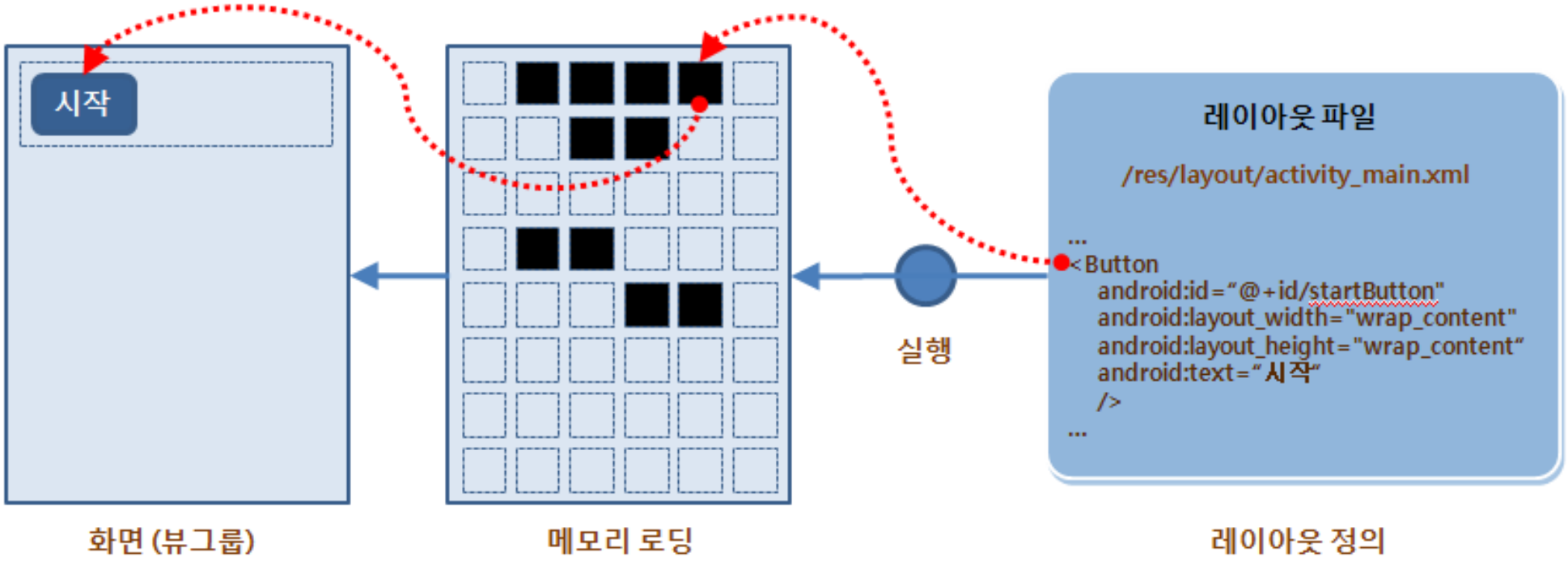


R.layout.레이아웃 파일 이름

인플레이션이란?

• XML 레이아웃에 정의된 뷰의 ID 속성 값을 이용하여 자바코드에서 메모리 상에 객체화된 뷰를 참조한다.
• XML 레이아웃 파일의 내용은 애플리케이션이 실행될 때 메모리로 로딩되어 객체화됨

- **인플레이션 (Inflation) : XML 레이아웃에 정의된 내용이 메모리에 객체화되는 과정**

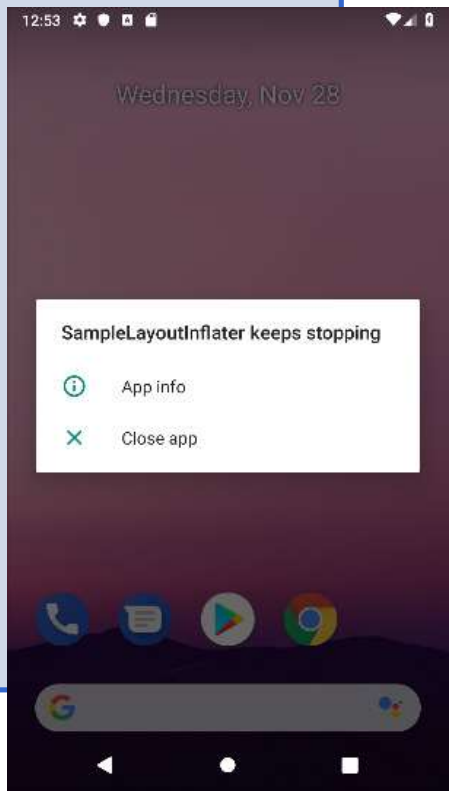


["시작" 버튼의 레이아웃 인플레이션 과정]

레이아웃 인플레이션의 이해 – 호출 순서

XML 레이아웃 파일은 프로젝트가 빌드되는 시점에 이진 파일로 컴파일되어 애플리케이션에 포함되긴 하지만 실행 시점이 되어서야 로드되어 메모리 상에 객체화됨

```
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);  //화면에 나타낼 뷰를 지정
  }    //activity_main.xml 에 정의된 뷰를 화면에 보여주라는 것
}
```



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

에러 발생 => 메모리 상에 객체화되지 않은 정보를 참조하려고 했기 때문

애플리케이션이 XML 레이아웃의 내용을 이해하도록 만드는 역할을 하는 것이 setContentView() 메소드

setContentView() 메소드의 역할

• setContentView() => 액티비티의 화면 전체를 설정하는 역할
• 화면 전체가 아닌 일부 뷰만을 위한 XML 레이아웃을 메모리상에 객체화하려면 별도의 인플레이션 객체를 사용해야 함

[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

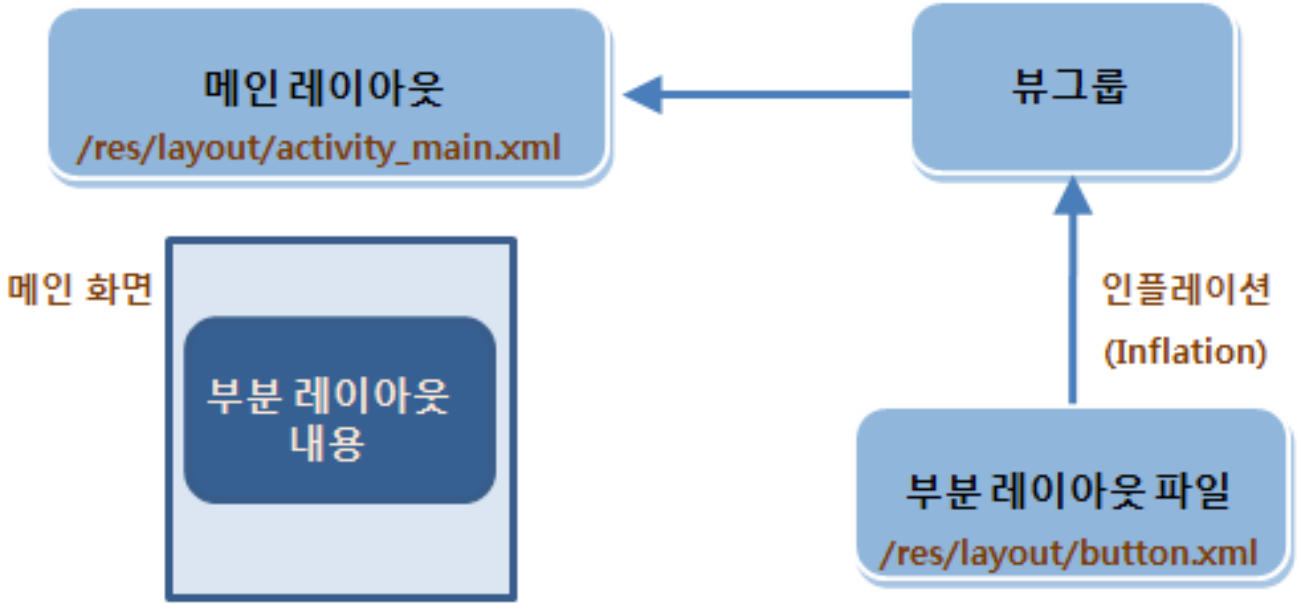
시스템 서비스로 제공되는 기능들은 모두 getSystemService() 메소드를 사용하여 객체를 참조한 후 사용해야 함

[Reference]

getSystemService(Context.LAYOUT_INFLATER_SERVICE)

- **전체 화면 중에서 일부분만을 차지하는 화면 구성요소들을 XML 레이아웃에서 로딩하여 보여줄 수 없을까?**
  - **LayoutInflater 라는 클래스를 제공**하며, 이 클래스는 시스템 서비스로 제공됨
  - getSystemService() 를 이용하여 LayoutInflater 객체를 참조한 후 사용

레이아웃 인플레이션의 개념도



[화면의 일부분을 XML 레이아웃 파일의 내용으로 적용하는 과정]

메인 레이아웃이 activity_main.xml 파일 안에 XML로 정의되어 있으면 setContentView() 코드를 통해 화면에 나타낼 수 있다. 그 중 일부를 분리하여 button.xml 이라는 파일에 정의하였다면 이 파일의 내용은 LayoutInflater 객체를 이용해 뷰그룹 객체로 객체화(인플레이션)한 후 메인 레이아웃에 추가되는 과정을 거치게 됨

화면 전체와 화면 일부

- **안드로이드에서 화면 : 소스와 화면 구성이 분리되어 있음**
  - 자바 소스 1개
  - XML 레이아웃 1개

- **화면 전체 : 액티비티 → setContentView 에서 인플레이션**
  - 액티비티를 위한 자바 소스 1개 : MainActivity.java
  - 액티비티를 위한 XML 레이아웃 1개 : activity_main.xml

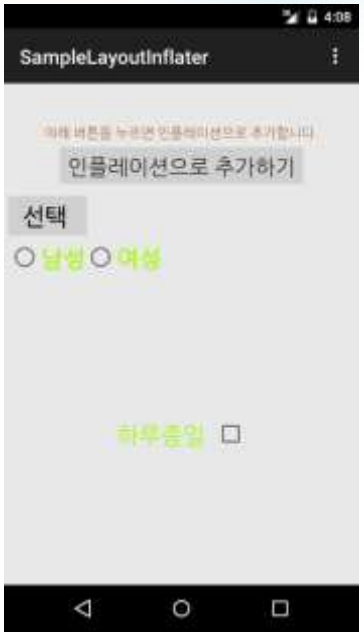
- **부분 화면 → 수동으로 인플레이션**
  - 부분화면을 위한 자바 소스 1개 또는 뷰 (뷰가 1개의 소스 파일로 분리될 수 있음)
  - 부분화면을 위한 XML 레이아웃 1개 : button.xml

레이아웃 인플레이션 예제

레이아웃 인플레이션 예제

-화면의 일부로 추가할 뷰의 XML 레이아웃 정의

-레이아웃 인플레이션 후 자바 코드에서 화면의 일부로 추가



메인 액티비티의 XML 레이아웃

-레이아웃 코드 작성

화면 일부의 XML 레이아웃

-레이아웃 코드 작성

메인 액티비티 코드

-메인 액티비티 코드 작성

메인 액티비티의 레이아웃

```
<LinearLayout
 android:id="@+id/contentsLayout"
 android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 >
</LinearLayout>
```

추가 레이아웃이 들어갈 위치

일부 화면을 위한 레이아웃

```
<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/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=" 선택 "
    android:textSize="24dp"
    android:textStyle="bold"
    android:gravity="center"
    />
```

1 버튼 정의

자바 코드 작성

button.xml에 정의된 레이아웃에 대한 인플레이션 수행

```
LinearLayout contentsLayout = (LinearLayout) findViewById(R.id.contentsLayout );
LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE );
inflater.inflate(R.layout.button, contentsLayout, true );
```

레이아웃 인플레이션 메소드

[Reference]

Xml 레이아웃 리소스를 지정하는 값, 뷰들을 객체화하여 추가할 대상이 되는 부모 컨테이너

View inflate (int resource, ViewGroup root, boolean attachToRoot)

[Reference] LayoutInflater 객체의 경우, 시스템 서비스로 제공되므로 getSystemService() 메소드를 이용해 객체를 참조하지만 LayoutInflater 클래스의 from() 메소드를 사용할 수도 있다

static LayoutInflater LayoutInflater.from (Context context)

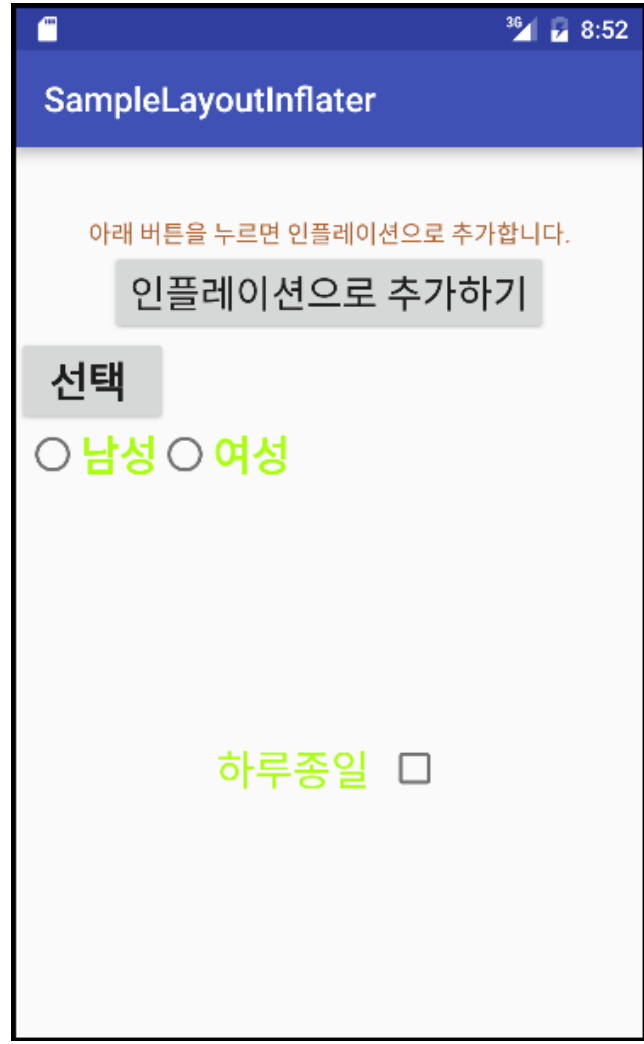
[Reference]

static View inflate (Context context, int resource, ViewGroup root)

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
        <TextView
                android:id="@+id/textView1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="40dp"
                android:gravity="center_horizontal"
                android:text="아래 버튼을 누르면 인플레이션으로 추가합니다."
                android:textSize="14dp"
                android:textColor="#ffad6535" />
        <Button
                android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                 android:layout_gravity="center_horizontal"
                android:text="인플레이션으로 추가하기"
                android:textSize="22dp"
                 android:onClick="onButton1Clicked" />

        <LinearLayout
                android:id="@+id/contentsLayout"
                 android:orientation="vertical"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
            >
        </LinearLayout>

</LinearLayout>
```

button.xml 에 정의된 내용을 추가할 부모 컨테이너



button.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
            <Button     android:id="@+id/selectButton"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="선택 "
                        android:textSize="24dp"
                        android:textStyle="bold"/>
            <RadioGroup
                        android:id="@+id/radioGroup01"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:orientation="horizontal"
                        android:paddingLeft="5dp"
                        android:paddingRight="5dp"          >
                        <RadioButton
                                    android:id="@+id/radio01"
                                    android:layout_width="wrap_content"
                                    android:layout_height="wrap_content"
                                    android:text="남성"
                                    android:textColor="#ffaaff10"
                                    android:textStyle="bold"
                                    android:textSize="24dp"                />
                        <RadioButton
                                    android:id="@+id/radio02"
                                    android:layout_width="wrap_content"
                                    android:layout_height="wrap_content"
                                    android:text="여성"
                                    android:textColor="#ffaaff10"
                                    android:textStyle="bold"
                                    android:textSize="24dp"/>
```

button.xml

```
            </RadioGroup>
            <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center_vertical|center_horizontal"
            android:paddingTop="10dp"
            >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="하루종일"
            android:textSize="24dp"
            android:paddingRight="10dp"
            android:textColor="#ffaaff10"
            />
        <CheckBox
            android:id="@+id/allDay"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           />
    </LinearLayout>
</LinearLayout>
```

```
/** 레이아웃 인플레이터를 이용해 레이아웃의 일부를 동적으로 로딩하는 방법에 대해 알 수 있다. */
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    /** 버튼을 눌렀을 때 레이아웃을 동적으로 로딩합니다. */
    public void onButton1Clicked(View v) {
        inflateLayout();
    }
    /** button.xml 에 정의된 레이아웃을 메인 액티비티의 레이아웃 일부로 추가하는 메소드 정의   */
    private void inflateLayout() {
        // XML 레이아웃에 정의된 contentsLayout 객체 참조 – 컨텐츠를 따로 추가할 레이아웃 객체 참조
        LinearLayout contentsLayout = (LinearLayout) findViewById(R.id.contentsLayout);

        // 인플레이션 수행
        LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);  //레이아웃 인플레이터 객체 참조
        inflater.inflate(R.layout.button, contentsLayout, true);  //button.xml 에 정의된 레이아웃에 대한 인플레이션 수행
        //=> contentsLayout 을 부모 컨테이너로 하여 button.xml 파일에 정의된 레이아웃을 추가하라

        // 새로 추가한 레이아웃 안에 들어있는 버튼 객체 참조
        Button btnSelect = (Button) findViewById(R.id.selectButton);
        final CheckBox allDay = (CheckBox) findViewById(R.id.allDay);  //체크박스 객체 참조
        btnSelect.setOnClickListener(new View.OnClickListener() {  //버튼을 누를때의 이벤트 처리
            public void onClick(View v) {
                if (allDay.isChecked()) {
                    allDay.setChecked(false);
                } else {
                    allDay.setChecked(true);
                }
            }
        });
    }
}
```

2. 화면 구성과 화면간 이동

액티비티

- 하나의 화면을 하나의 액티비티로 생각할 수 있다
- 애플리케이션을 구성하는 화면을 액티비티로 구현하고 각각의 화면 간에 이동하는 과정은 각각의 액티비티를 필요에 따라 띄우거나 닫는 과정과 같다
- 액티비티 – 애플리케이션을 구성하는 네 가지 기본 요소 중의 하나로 대부분의 애플리케이션이 적어도 하나 이상 포함하고 있다.
- 안드로이드 애플리케이션의 기본 구성 요소 네 가지
  - 액티비티, 서비스, 브로드캐스트 수신자(Broadcast Receiver), 내용 제공자(Content Provider)

액티비티

- 애플리케이션 구성요소들은 애플리케이션이 만들어져 설치되면 안드로이드 시스템이 이들에 대한 정보를 요구한다
- 애플리케이션을 구성하는 구성요소 네 가지는 새로 만들 때마다 항상 그 정보를 매니페스트 파일에 추가해야 하며, 이를 통해 애플리케이션을 구성하고 있는 정보를 시스템에 알려줄 수 있다.
- 액티비티를 띄우기 위해 사용하는 메소드

- startActivity() : 단순히 액티비티를 띄워 화면에 보이도록 함
- startActivityForResult() : 어떤 액티비티를 띄운 것인지, 띄웠던 액티비티로부터의 응답을 받아 처리하는 코드가 필요한 경우 사용

[Reference]

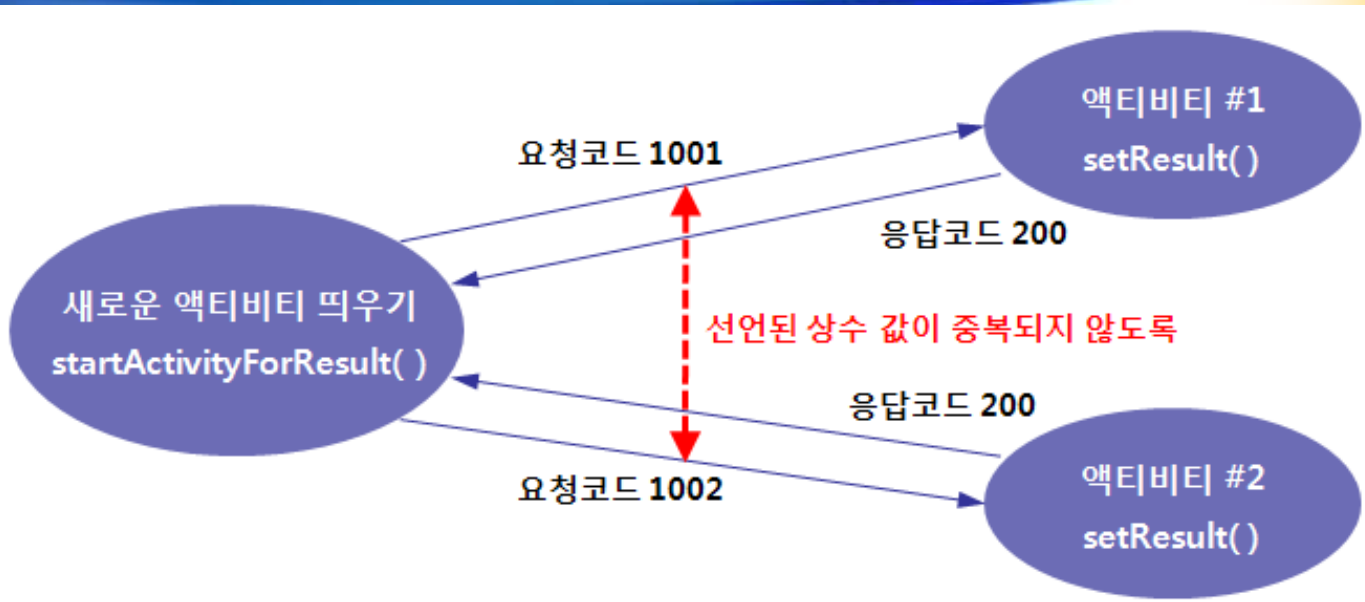
정수 코드값 – 각각의 액티비티를 구분하기 위해 사용

startActivityForResult(Intent intent, int requestCode)

하나의 액티비티에서 다른 액티비티를 띄우기만 하는 것은 상관없지만 띄웠던 액티비티에서 원래의 액티비티로 응답을 보내온다면 새로 띄웠던 여러 액티비티 중에 어떤 것으로부터 온 응답인지 구분할 필요가 있기 때문에 이 메소드가 사용됨

[안드로이드 애플리케이션을 구성하는 네 가지 구성요소]

화면 구성과 화면간 이동 과정



[액티비티에 선언된 상수 값 중복에 주의]

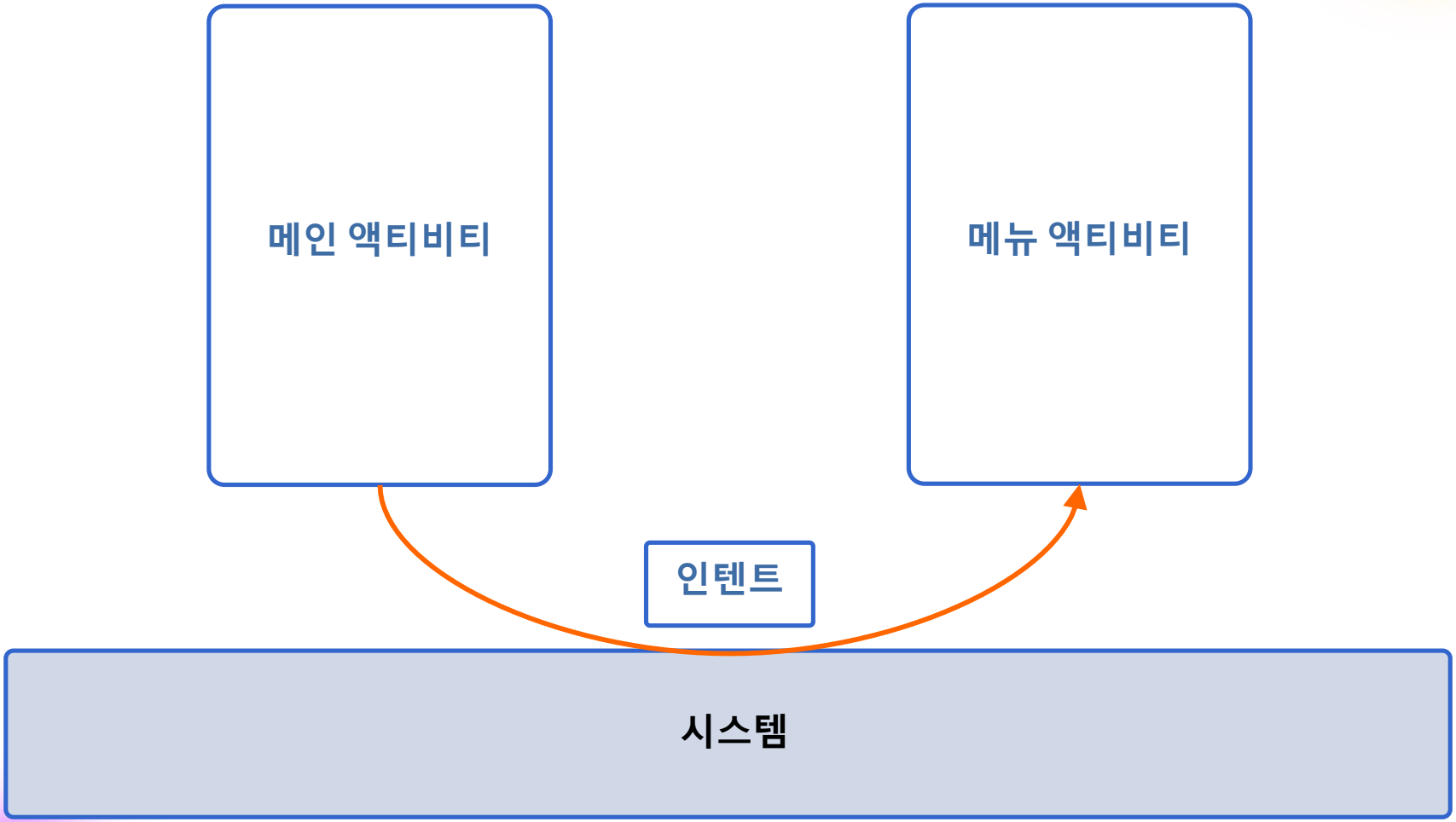
[Reference]

•파라미터1 - 액티비티를 띄울 때 전달했던 코드와 같다 => 어떤 액티비티로부터 응답을 받은 것인지 구분할 때 사용
•파라미터 2- 응답을 보내 온 액티비티로부터 전달된 값, 임의의 코드이긴 하지만 성공(200), 실패(400)와 같이 액티비티의 기능을 수행한 결과 값을 전달
•파라미터 3 – 응답을 보내 온 액티비티로부터 전달한 인텐트, 필요한 데이터가 있을 때 인텐트에 데이터를 넣어 전달

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

띄웠던 액티비티가 응답을 보내오면 그 응답을 처리하는 역할을 함

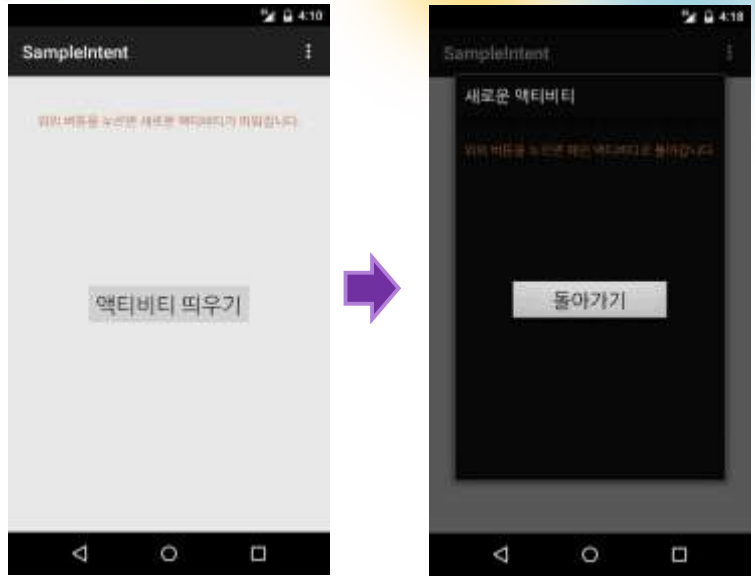
화면을 띄울 때 시스템으로 요청하기



인텐트 객체는 액티비티를 띄우기 위한 목적으로 사용되며
액티비티간에 데이터를 전달하는 데에도 사용될 수 있다

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;
 ...
 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 );
     startActivityForResult(intent, REQUEST_CODE_ANOTHER );
  }
 });
```

1 다른 액티비티를 띄우기 위한 요청코드 정의

2 또 다른 액티비티를 띄우기 위한 인텐트 객체 생성

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 (getBaseContext(),
                    "onActivityResult called with code : " + resultCode,
                    Toast.LENGTH_LONG);
     toast.show();

     if (resultCode == 1) {
       String name = data.getExtras().getString("name");
       toast = Toast.makeText(getBaseContext(), "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

setResult() – 현재 액티비티를 띄운 액티비티로 응답을 보낼 때 사용됨

인텐트 객체에 데이터를 넣는 간단한 방법
- 문자열을 putExtra() 메소드로 설정, (key, value) 쌍으로 넣어 주어야 함

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">
   <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
  </activity>
  <activity android:name=  ".AnotherActivity"
            android:label="@string/app_name">
  </activity>
 </application>
 <uses-sdk android:minSdkVersion="7" />
</manifest>
```

1 또 다른 액티비티를 등록한 태그

애플리케이션 실행 화면


4:10
SampleIntent
위의 버튼을 누르면 새로운 액티비티가 띄워집니다.
액티비티 띄우기


4:18
SampleIntent
새로운 액티비티
위의 버튼을 누르면 메인 액티비티로 돌아갑니다.
돌아가기


4:19
SampleIntent
위의 버튼을 누르면 새로운 액티비티가 띄워집니다.
액티비티 띄우기
onActivityResult() 메소드가 호출됨. 요청 코드 : 1001, 결과코드 : -1

액티비티 구성 과정 정리

(1) 새로운 액티비티의 XML 레이아웃 정의

(2) 새로운 액티비티 코드 작성

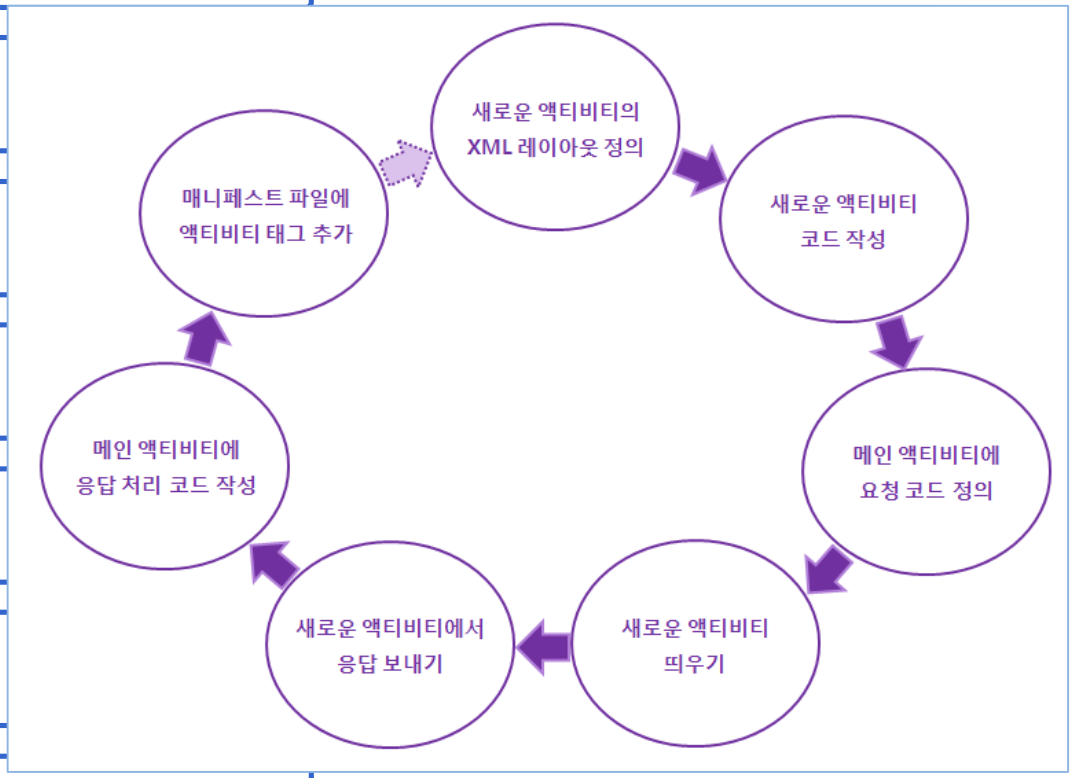
(3) 메인 액티비티에 요청 코드 정의

(4) 새로운 액티비티 띄우기

(5) 새로운 액티비티에서 응답 보내기

(6) 메인 액티비티에 응답 처리 코드 작성

(7) 매니페스트 파일에 액티비티 태그 추가



[액티비티 추가와 요청 그리고 응답 과정]

MainActivity

```
/** 인텐트를 이용해 새로운 액티비티를 띄우고 다시 돌아오는 방법에 대해 알 수 있다. */
public class MainActivity extends AppCompatActivity {
    /** 요청 코드 정의    */
    public static final int REQUEST_CODE_ANOTHER = 1001;

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

    public void onButton1Clicked(View v) {
        // 인텐트 객체를 만드는 방법 #1

        // 인텐트 객체를 만듭니다.
        Intent intent = new Intent(getBaseContext(), AnotherActivity.class);

        // 액티비티를 띄워주도록 startActivityForResult() 메소드를 호출합니다.
        startActivityForResult(intent, REQUEST_CODE_ANOTHER);

        // 인텐트 객체를 만드는 방법 #2
        //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);

    }
```

MainActivity

```
    /**
     * 새로운 액티비티에서 돌아올 때 자동 호출되는 메소드
     */
    protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
        super.onActivityResult(requestCode, resultCode, intent);

        if (requestCode == REQUEST_CODE_ANOTHER) {
            Toast toast = Toast.makeText(getBaseContext(), "onActivityResult() 메소드가 호출됨. 요청코드 : " + requestCode + ",
결과코드 : " + resultCode, Toast.LENGTH_LONG);
            toast.show();

            if (resultCode == RESULT_OK) {
                String name = intent.getExtras().getString("name");
                toast = Toast.makeText(getBaseContext(), "응답으로 전달된 name : " + name, Toast.LENGTH_LONG);
                toast.show();
            }
        }

    }

}
```

AnotherActivity

```
/**
 * 새로운 액티비티
*/
public class AnotherActivity extends Activity {
    Button backButton;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.another);

        backButton = (Button) findViewById(R.id.backButton);

        // 버튼을 눌렀을 때 메인 액티비티로 돌아갑니다.
        backButton.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        // 객체를 만듭니다.
                        Intent resultIntent = new Intent();
                        resultIntent.putExtra("name", "mike");

                        // 응답을 전달하고 이 액티비티를 종료합니다.
                        setResult(RESULT_OK, resultIntent);
                        finish();
                }
        });

    }

}
```

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

        <Button
                android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true"
                android:text="액티비티 띄우기"
                android:textSize="24dp"
                android:onClick="onButton1Clicked"/>

        <TextView
                android:id="@+id/textView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/button1"
                android:layout_marginTop="40dp"
                android:gravity="center_horizontal"
                android:text="위의 버튼을 누르면 새로운 액티비티가 띄워집니다."
                android:textSize="14dp"
                android:textColor="#ffad6535" />

</RelativeLayout>
```

another.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="300dp"
    android:layout_height="400dp" >

        <Button
                android:id="@+id/backButton"
                android:layout_width="180dp"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true"
                android:text="돌아가기"
                android:textSize="22dp"
                 />

        <TextView
                android:id="@+id/textView1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/backButton"
                android:layout_marginTop="20dp"
                android:gravity="center_horizontal"
                android:text="위의 버튼을 누르면 메인 액티비티로 돌아갑니다."
                android:textSize="14dp"
                android:textColor="#ffad6535" />

</RelativeLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.androidtown.intent.basic">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".AnotherActivity"
            android:label="@string/title_another"
            android:theme="@android:style/Theme.Dialog"
            >
        </activity>

    </application>

</manifest>
```

상단의 제목 부분 글자 지정

화면이 대화상자 모양으로 보이도록 android:theme 속성 지정

3. 인텐트와 데이터 전달

인텐트와 데이터 전달

- 인텐트

– 다른 액티비티를 띄우거나 기능을 동작시키기 위한 수단으로 사용됨

- 무언가 작업을 수행하기 위해 사용되는 일종의 전달 수단

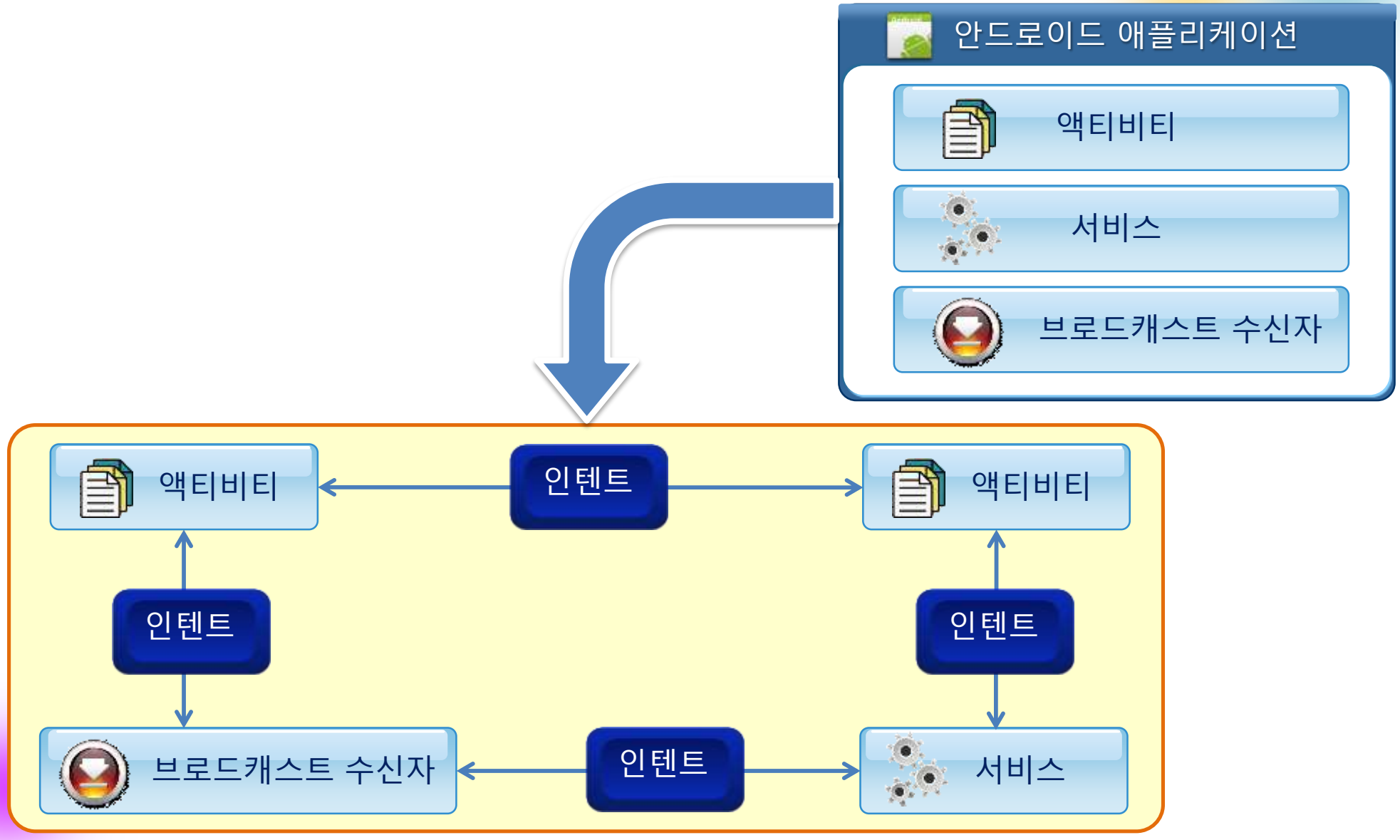
- 다른 애플리케이션의 기능을 수행하는 등 훨씬 유연한 기능의 애플리케이션을 만들 수 있다

- 애플리케이션 구성 요소 간에 작업 수행을 위한 정보를 전달하는 역할

- 인텐트를 정의한 패키지 – android.content.Intent
- 다른 애플리케이션 구성요소에 인텐트를 전달할 수 있는 메소드

startActivity() – 액티비티를 화면에 띄울 때 사용
startService() 또는 bindService() – 서비스를 시작할 때
broadcastIntent() – 브로드캐스팅을 수행할 때 사용

인텐트와 데이터 전달


안드로이드 애플리케이션
액티비티
서비스
브로드캐스트 수신자
액티비티
인텐트
액티비티
인텐트
인텐트
브로드캐스트 수신자
인텐트
서비스

인텐트의 기본 구성 요소

- 액션(Action) : 수행할 기능

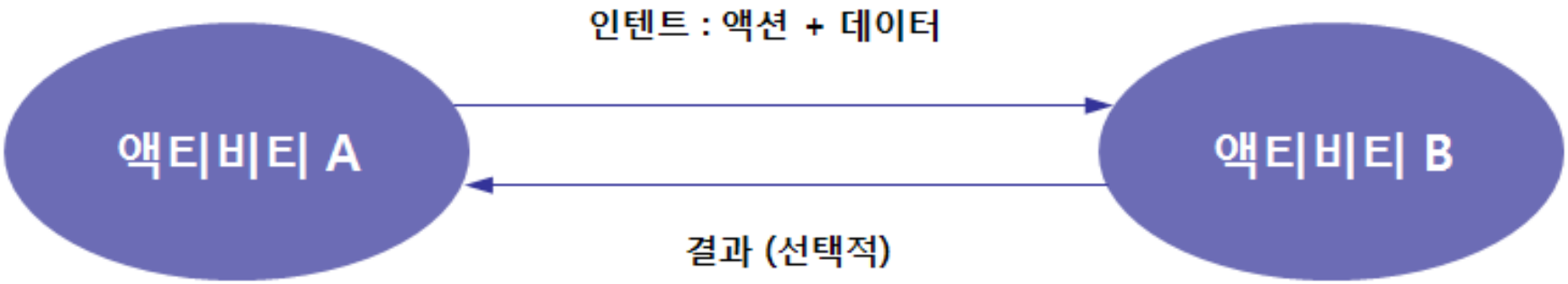
  예) ACTION_VIEW(웹페이지 주소를 이용해 브라우저를 띄울 때 사용), ACTION_EDIT

- 데이터(Data) : 액션이 수행될 대상 데이터

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

인텐트에 포함되어 있는 데이터는 그 유형에 따라 시스템에서 런타임시 자동으로 해당 액티비티를 찾아 띄워주는데 사용됨

그 유형은 등록된 MIME 타입으로 구분

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

  - MIME 타입에 따라 안드로이드 시스템에서 적절한 다른 애플리케이션의 액티비티를 찾은 후 띄우는 방식을 사용

  - 속성 : 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

예제

- 인텐트를 이용하는 대표적인 두 가지 경우

[1] 인텐트에 액션과 데이터를 넣어 다른 애플리케이션의 액티비티를 띄우는 경우

[2] 컴포넌트 이름을 이용해 새로운 액티비티를 띄우는 경우

예제1 - 인텐트에 액션과 데이터를 넣어 다른 애플리케이션의 액티비티를 띄우는 경우

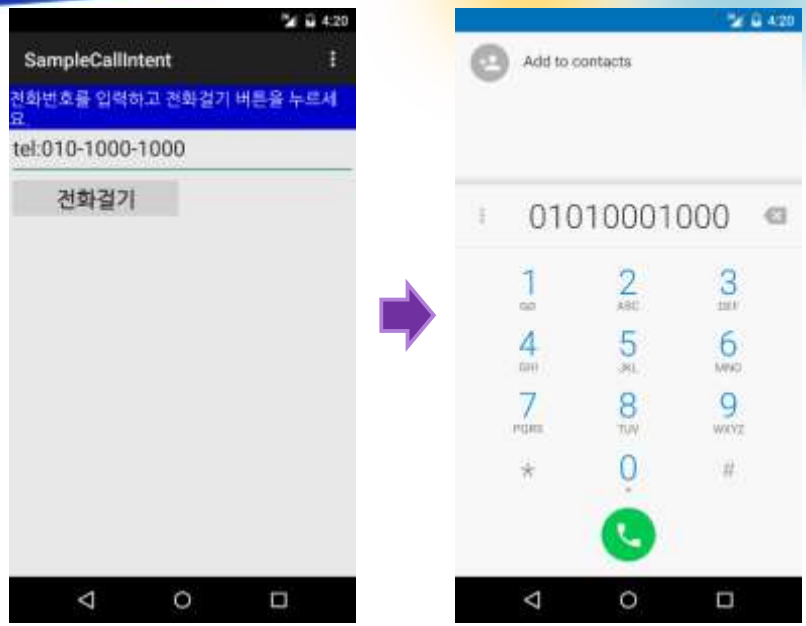
사용자가 전화번호를 입력하고 버튼을 누르면 그 전화번호로 전화를 걸 수 있도록 인텐트 액션을 적용

데이터 전달 예제

데이터 전달 예제

-액티비티 간에 인텐트로 데이터 전달

-전화걸기 기능 이용



XML 레이아웃 정의

-전화걸기 화면을 띄우는 화면 구성

메인 액티비티 코드 작성

-전화걸기 화면을 띄울 때 전화번호 전달

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"     android:layout_height="match_parent"
    android:orientation="vertical" >
            <TextView
                    android:id="@+id/textView1"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:background="#ff0000cc"
                    android:text="전화번호를 입력하고 전화걸기 버튼을 누르세요."
                    android:textSize="18dp"
                    android:textColor="#ffffffff"
                    />
            <EditText
                    android:id="@+id/editText1"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:text="tel:010-1000-1000"
                    android:textSize="22dp"
                    />
            <Button
                    android:id="@+id/button1"
                    android:layout_width="180dp"
                    android:layout_height="wrap_content"
                    android:text="전화걸기"
                    android:textSize="22dp"
                    android:textStyle="bold"
                    android:onClick="onButton1Clicked"
                    />

</LinearLayout>
```

MainActivity.java

```
package org.androidtown.intent.call;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
/** * 인텐트로 전화걸기 화면을 보여주는 방법을 알 수 있다. */
public class MainActivity extends AppCompatActivity {
    TextView textView1;
    EditText editText1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textView1 = (TextView) findViewById(R.id.textView1);
        editText1 = (EditText) findViewById(R.id.editText1);
    }

    public void onButton1Clicked(View v) {
        // 입력상자에 입력한 전화번호를 가져옴
        String data = editText1.getText().toString();

        // 인텐트를 만들고 이것을 이용해 액티비티를 띄워줌
        Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse(data));
        startActivity(intent);
    }
}
```

예제2 - 컴포넌트 이름을 이용해 새로운 액티비티를 띄우는 경우

```
/** * 인텐트를 이용해 새로운 액티비티를 띄우고 다시 돌아오는 방법에 대해 알 수 있다. */
public class MainActivity extends AppCompatActivity {
    /**   * 요청 코드 정의    */
    public static final int REQUEST_CODE_ANOTHER = 1001;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onButton1Clicked(View v) {
        // 인텐트 객체를 만드는 방법 #2
        Intent intent = new Intent();
        ComponentName name = new ComponentName("com.example.inflateapp",
                "com.example.inflateapp.AnotherActivity"); //컴포넌트 이름을 지정할 수 있는 객체 생성
        intent.setComponent(name);    //인텐트 객체에 컴포넌트 지정

        startActivityForResult(intent, REQUEST_CODE_ANOTHER);   //액티비티 띄우기
    }

    /**     * 새로운 액티비티에서 돌아올 때 자동 호출되는 메소드     */
    protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
        super.onActivityResult(requestCode, resultCode, intent);
        if (requestCode == REQUEST_CODE_ANOTHER) {
            Toast toast = Toast.makeText(getBaseContext(), "onActivityResult() 메소드가 호출됨. 요청코드 : " +
requestCode + ", 결과코드 : " + resultCode, Toast.LENGTH_LONG);
            toast.show();
            if (resultCode == RESULT_OK) {
                String name = intent.getExtras().getString("name");
                toast = Toast.makeText(getBaseContext(), "응답으로 전달된 name : " + name, Toast.LENGTH_LONG);
                toast.show();
            }        }    } }
```

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

Continued..

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

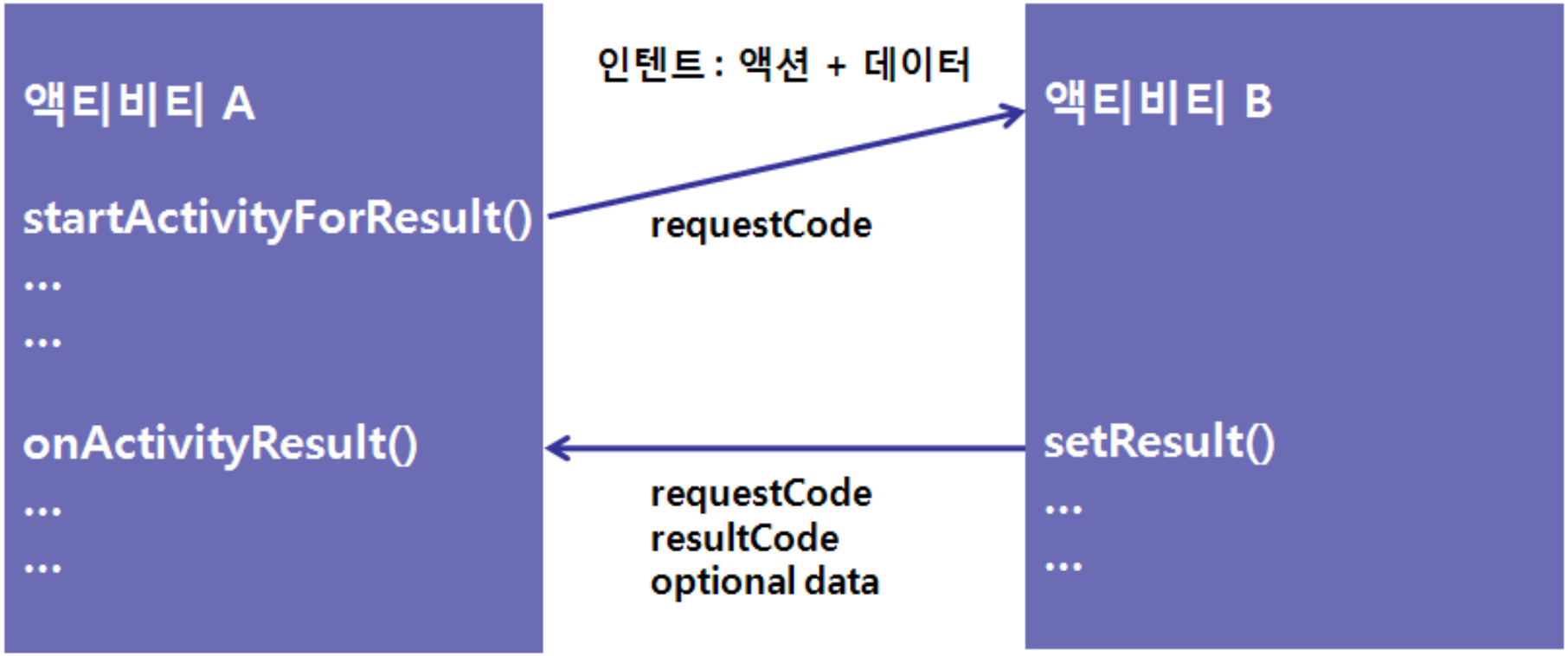
메인 액티비티 코드 만들기

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

```
Intent intent = new Intent();
ComponentName name = new ComponentName("com.example.inflateapp",
                        "com.example.inflateapp.AnotherActivity" );
intent.setComponent(name);
startActivityForResult(intent, REQUEST_CODE_ANOTHER );
```

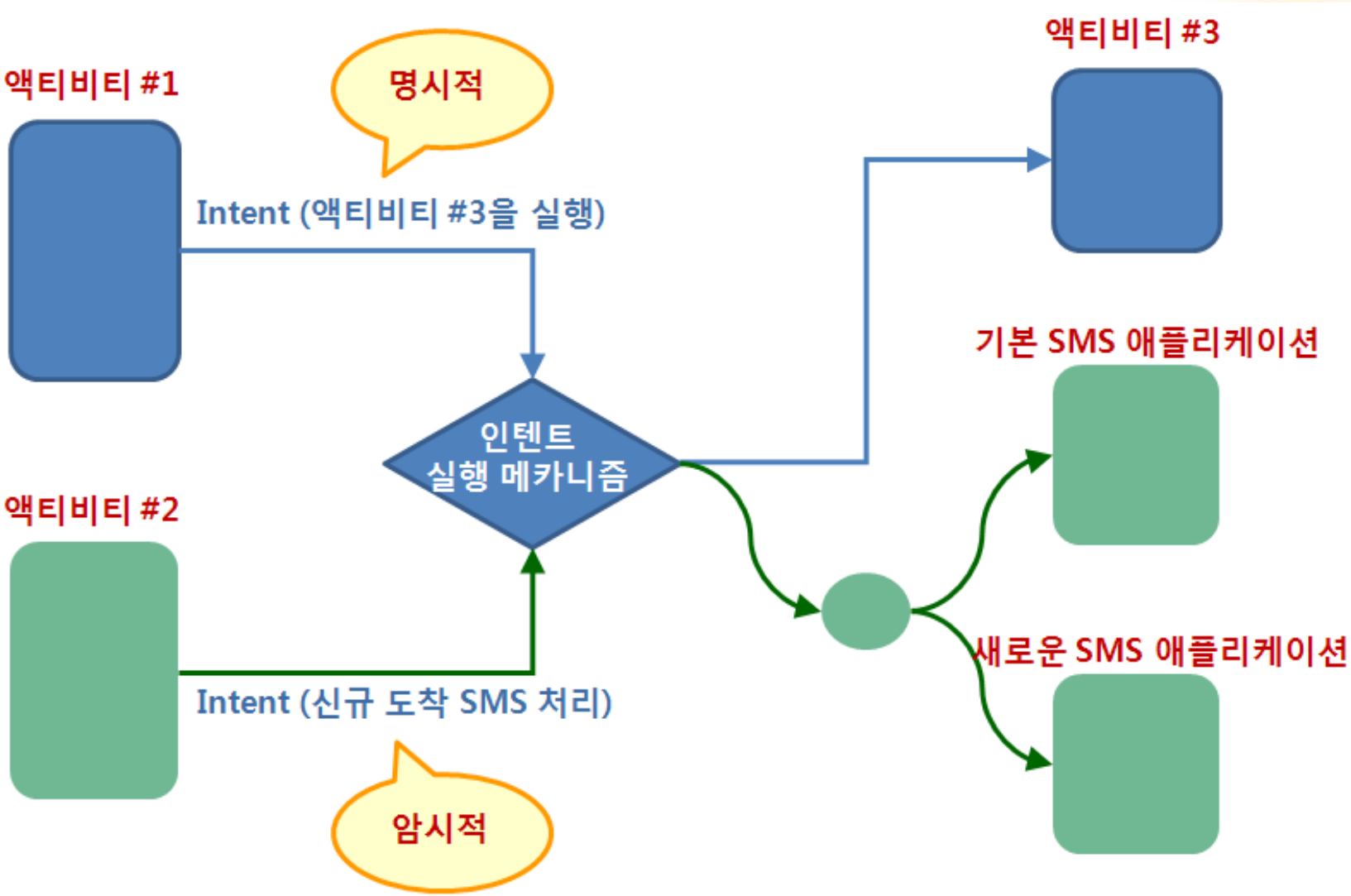
액티비티 간

- 액티비티 B가 종료되기 전에 setResult() 메소드를 호출하면 액티비티 A에게 응답을 보낼 수 있다
- 결과값 코드 – 임의의 정수나 액티비티 상수로 미리 정의된 코드
  - 성공을 의미하는 상수 – Activity.RESULT_OK,
  - 실패 상수 - Activity.RESULT_CANCELED
- 액티비티 B에서 보낸 응답은 부모 액티비티의 onActivityResult() 메소드에서 받을 수 있다.
- 액티비티가 비정상종료되면 Activity.RESULT_CANCELED값이 전달됨



startActivityForResult() 메소드를 이용한 액티비티 간의 전환

인텐트를 해석하는 과정


액티비티 #1
명시적
Intent (액티비티 #3을 실행)
액티비티 #3
인텐트
실행 메카니즘
기본 SMS 애플리케이션
액티비티 #2
새로운 SMS 애플리케이션
Intent (신규 도착 SMS 처리)
암시적

인텐트를 해석하는 과정

- 액티비티 #1은 명시적 인텐트를 이용해 액티비티 #3을 실행시킴
- 액티비티 #3은 문자 메시지를 받아 처리하는 애플리케이션으로 액티비티 #3의 컴포넌트 이름을 지정하면 해당 애플리케이션을 실행할 수 있으므로 액티비티 #3을 직접 호출하여 문자메시지를 처리함

- 액티비티 #2는 암시적 인텐트를 이용해 액티비티 #3을 실행시킴
- 액티비티 #3은 지정한 애플리케이션이 아니라 문자메시지를 처리할 수 있는 여러 애플리케이션들 중의 하나일 수 있으므로 각각 인텐트 필터를 이용해 시스템이 인텐트를 비교할 수 있도록 만들어줌

- 시스템이 요청하는 인텐트의 정보를 받아 처리할 애플리케이션 구성요소를 찾기 위해 필요한 정보가 인텐트 필터
- 인텐트 필터는 인텐트가 가지는 액션 정보를 동일하게 가질 수 있는데 이 인텐트를 해석하는 메커니즘은 기본적으로 전달하고자 하는 대상 인텐트와 설치된 애플리케이션들이 가지는 인텐트 필터를 서로 비교하는 방식을 취함

플래그

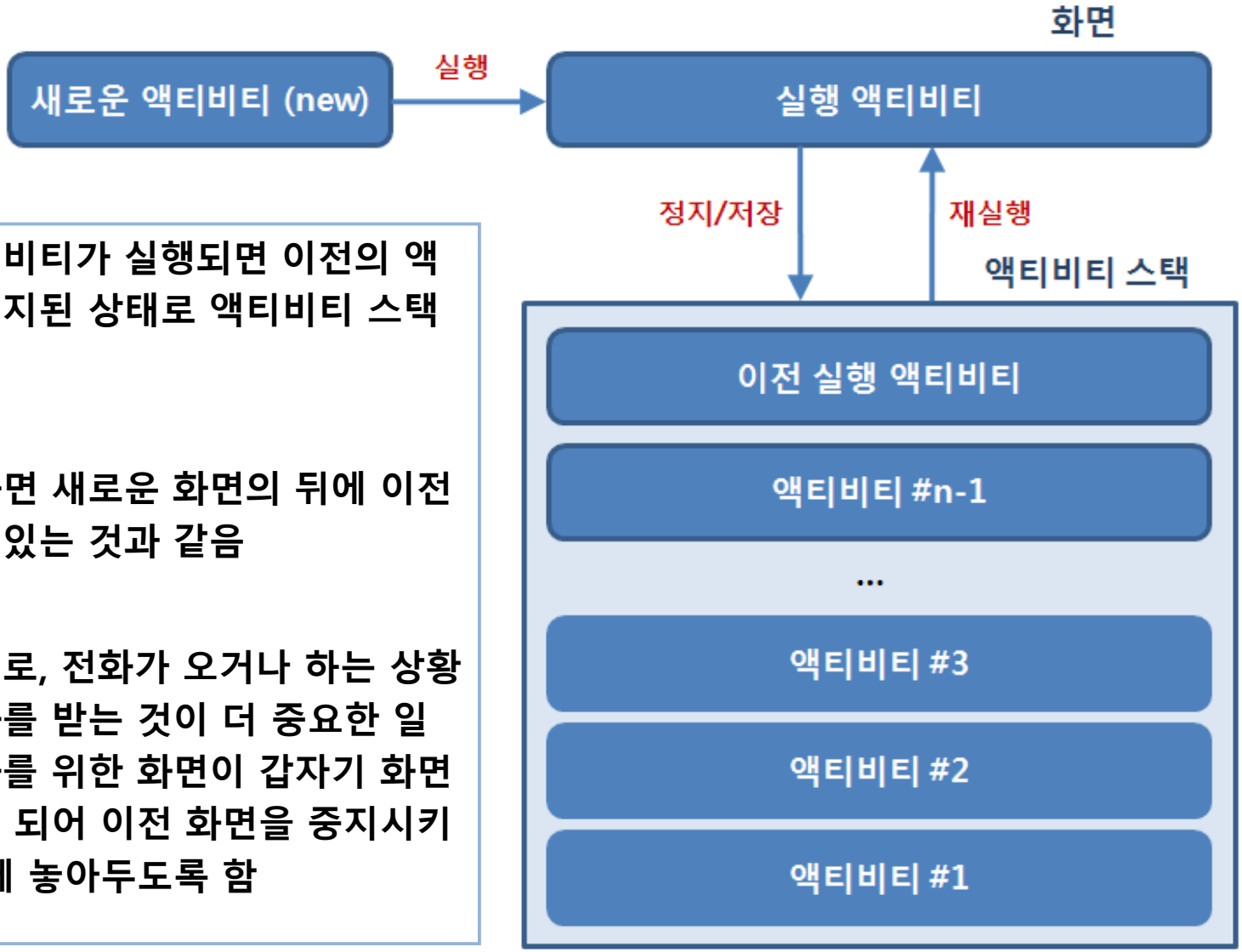
플래그(Flag)

- 액티비티가 동작하는 방식을 설정할 수 있도록 해줌

액티비티가 처리되는 방식

- 액티비티는 **액티비티 스택으로 관리**됨
- 이 스택은 액티비티를 차곡차곡 쌓아두었다가 가장 상위에 있던 액티비티가 없어지면 이전의 액티비티를 다시 실행하도록 도와줌
- 새로운 액티비티를 만들어 매니테스트 파일에 등록하면 그액티비티는 startActivity() 메소드를 이용해 실행될 수 있다.
- 이렇게 실행된 액티비티는 화면에 띄워지는데 **새로운 액티비티가 화면에 띄워지게 되면 이전에 있던 액티비티는 액티비티 스택에 저장**되고 새로운 액티비티가 화면에 보이는 구조임
- **화면에 보이던 액티비티가 없어지면 액티비티 스택의 가장 위에 있던 액티비티가 화면상에 보이면서 재실행**됨
- 동일한 액티비티를 여러 번 실행한다면 동일한 액티비티가 여러 개 스택에 들어가 있고 동시에 데이터를 여러 번 접근하거나 리소스를 여러 번 사용하는 문제가 발생할 수 있다 => 이러한 문제들을 해결할 수 있도록 도와주는 것이 플래그

액티비티 스택



- 새로운 액티비티가 실행되면 이전의 액티비티는 정지된 상태로 액티비티 스택에 들어감
- 쉽게 생각하면 새로운 화면의 뒤에 이전 화면이 숨어있는 것과 같음
- 구체적인 예로, 전화가 오거나 하는 상황에서는 전화를 받는 것이 더 중요한 일이므로 통화를 위한 화면이 갑자기 화면을 점유하게 되어 이전 화면을 중지시키고 화면 뒤에 놓아두도록 함

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

[FLAG_ACTIVITY_SINGLE_TOP 플래그를 사용한 경우]

액티비티를 생성할 때 이미 생성된 액티비티가 있으면 그 액티비티를 그대로 사용하라는 플래그

액티비티 플래그 사용 예

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

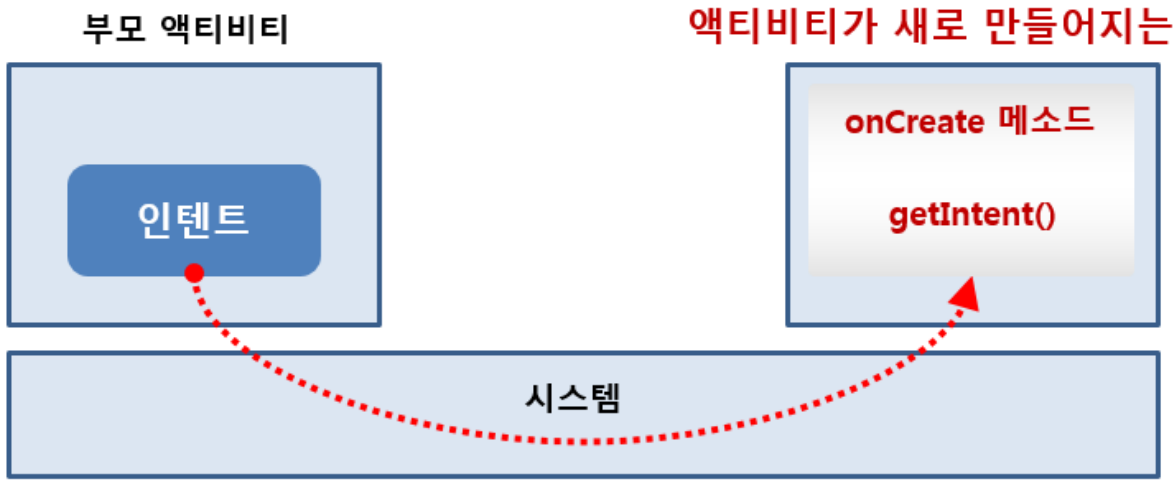
1 인텐트 객체 생성

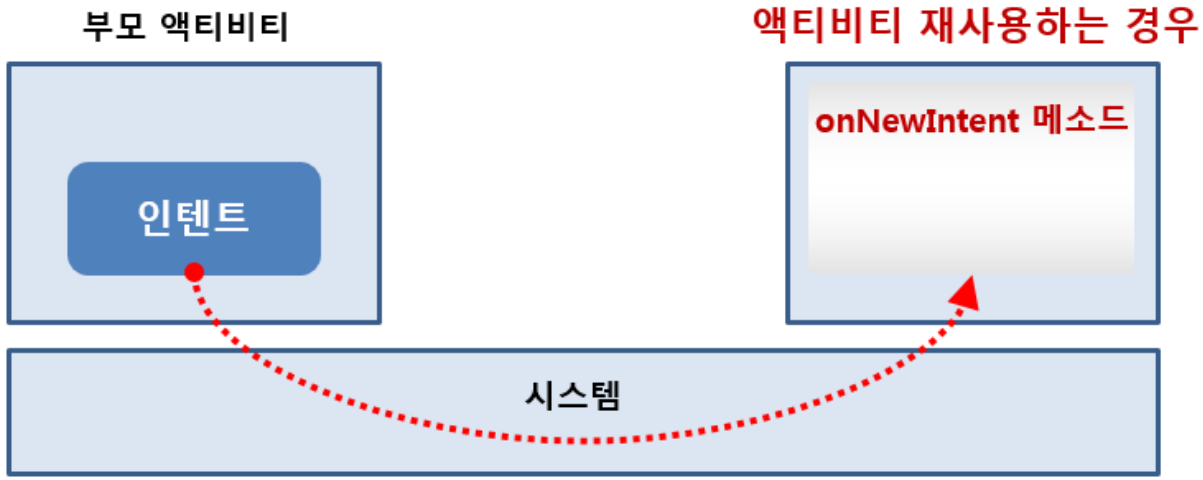
2 부가 데이터 넣기

3 인텐트 플래그 설정

4 인텐트 띄우기

액티비티에서 인텐트를 전달받는 두 가지 경우





다른 액티비티 플래그의 사용

[FLAG_ACTIVITY_NO_HISTORY 플래그를 사용한 경우]

처음 이후에 실행된 액티비티는 액티비티 스택에 추가되지 않는다. 즉, 플래그가 설정되어 있지 않은 경우에는 이전에 실행되었던 액티비티가 스택에 추가되므로 [Back]키를 누르면 이전의 액티비티가 보이게 되지만 이 플래그를 사용하면 항상 맨 처음에 실행되었던 액티비티가 바로 보이게 됨

[FLAG_ACTIVITY_CLEAR_TOP 플래그를 사용한 경우]

이 액티비티 위에 있는 다른 액티비티를 모두 종료시키게 됨
홈 화면과 같이 다른 액티비티보다 항상 우선하는 액티비티를 만들 때 사용

부가 데이터 전달하기

[Reference]

Intent putExtra(String name, String value)
Intent putExtra(String name, int value)
Intent putExtra(String name, boolean value)

String getStringExtra(String name)
int getIntExtra(String name, int defaultValue)
boolean getBooleanExtra(String name, boolean defaultValue)

[Reference]

public abstract int describeContents()
public abstract void writeToParcel(Parcel dest, int flags)

- 화면과 화면 간에 데이터를 전달하고 싶다면 인텐트의 부가 데이터(Extra)로 넣어 전달하는 방법을 사용함

- 인텐트는 애플리케이션 구성 요소 간에 데이터를 전달하는 방법을 제공하는 것이므로 화면과 화면 간 뿐만 아니라 화면과 서비스 간, 또는 브로드캐스트 수신자와 화면 간 등등 애플리케이션 구성 요소 간에 부가 데이터로 넣어 데이터를 전달할 수 있음

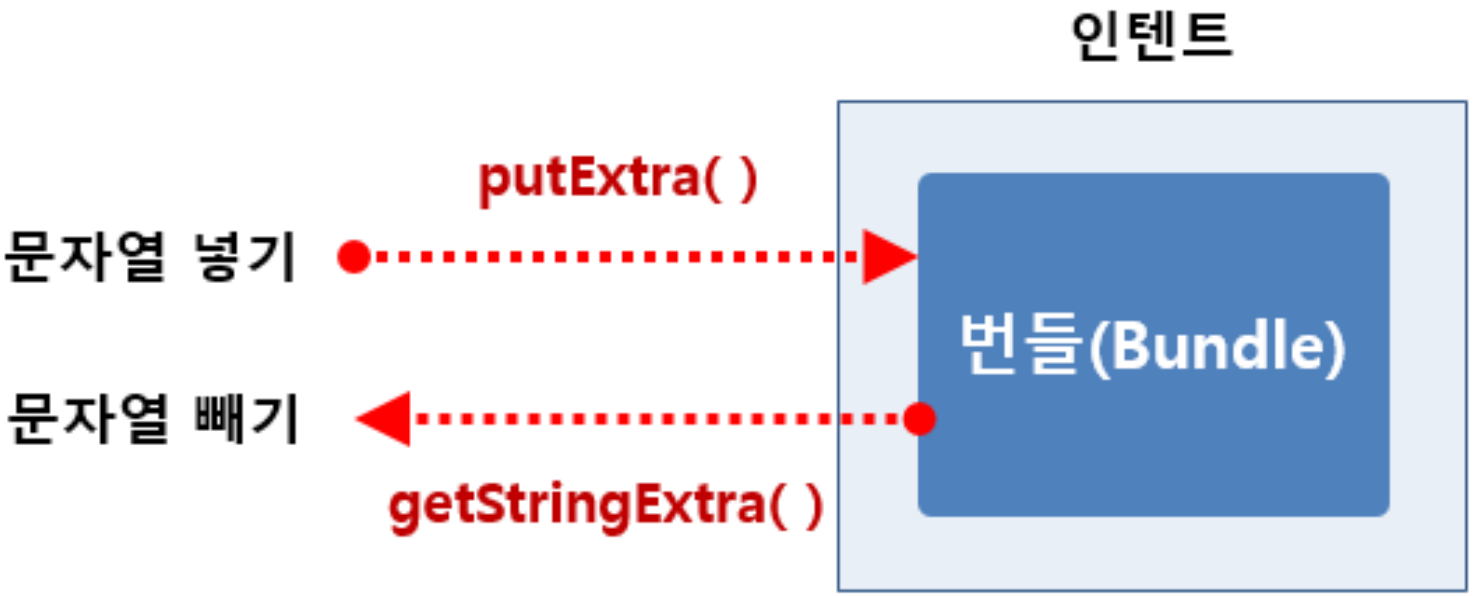
부가 데이터

- 액티비티 간에 데이터를 전달하기 위해 사용하는 번들 객체는 인텐트 안에 들어 있기 때문에 putXXX()와 getXXX() 메소드를 이용해 데이터를 넣거나 볼 수 있다
- 기본적으로 기본 데이터 타입만을 지원 => 문자열이나 정수와 같은 데이터를 키와 데이터 값의 쌍으로 만들어 넣게 됨
- 문자열을 직접 넣는 경우에는 putExtra(), getStringExtra() 메소드 사용
- getXXX() 메소드 – 데이터 값이 없으면 디폴트로 설정된 값이 리턴됨
- 전달하고 싶은 데이터가 기본 데이터 타입이 아닌 객체인 경우에는 객체 자체를 전달할 수 없다

- 객체의 데이터들을 바이트 배열로 변환하여 전달하거나
- Serializable 인터페이스를 구현하는 객체를 만들어 직렬화한 후 전달
- 안드로이드에서는 Serializable 인터페이스와 유사한 Parcelable 인터페이스를 권장함 <= 이 인터페이스를 이용해 내부적인 데이터 전달 메커니즘이 만들어져 있기 때문

  - 이 인터페이스를 이용하면 데이터를 전달하기 위해 객체를 직접 번들에 추가할 수 있다.

번들 객체



부가 데이터

Parcelable 인터페이스를 구현하여 객체를 직접 전달하려면 다음 두 메소드를 구현해야 함

public abstract int describeContents()

public abstract void writeToParcel(Parcel dest, int flags) – 객체가 가지고 있는 데이터를 Parcel 객체로 만들어주는 역할

- Parcel 객체는 Bundle 객체처럼 readXXX(), writeXXX() 메소드를 제공 => 기본 데이터 타입을 넣고 확인할 수 있다
- 위의 두 가지 메소드를 구현하였다면 CREATOR 라는 상수를 만들어야 함
- 이 상수는 Parcel 객체로부터 데이터를 읽어 들여 객체를 생성하는 역할을 함

Parcelable 예제

Parcelable 예제

-Parcelable 객체 만들기

-또다른 액티비티에 Parcelable 객체 전달하기

SimpleData 객체 정의

-Parcelable 인터페이스를 구현하는 객체 정의

메인 액티비티 코드 작성

-또 다른 액티비티를 띄워줄 때 부가데이터 전달

또다른 액티비티 코드 작성

-전달된 객체를 이용해 데이터 표시

```
/** 인텐트를 이용해 전달할 때 Parcelable 객체로 만들어 전달하는 방법에 대해 알 수 있다. */
public class MainActivity extends AppCompatActivity {
    public static final int REQUEST_CODE_ANOTHER = 1001;
    /** 부가 데이터의 키 값 정의      */
    public static final String KEY_SIMPLE_DATA = "data";

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

    public void onButton1Clicked(View v) {
        // 인텐트 객체를 만듭니다.
        Intent intent = new Intent(this, AnotherActivity.class);

        SimpleData data = new SimpleData(100, "Hello Android!");
        intent.putExtra(KEY_SIMPLE_DATA, data);

        // 액티비티를 띄워주도록 startActivityForResult() 메소드를 호출합니다.
        startActivityForResult(intent, REQUEST_CODE_ANOTHER);
    }

   protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST_CODE_ANOTHER) {
            Toast toast = Toast.makeText(this, "onActivityResult() 메소드가 호출됨. 요청코드 : " + requestCode + ", 결과코드 : "
+ resultCode, Toast.LENGTH_LONG);
            toast.show();
        }
    } }
```

- MainActivity 코드에서는 인텐트에 부가 데이터를 넣을 때 SimpleData 객체를 하나 만든 후 putExtra() 메서드를 이용해 부가 데이터를 추가
- SimpleData 객체는 Parcelable 인터페이스를 구현하도록 정의한 것

```
package org.androidtown.basic.parcelable;
import android.os.Parcel;
import android.os.Parcelable;
/**Parcelable 인터페이스를 구현하는 클래스 정의 */
public class SimpleData implements Parcelable {
        // 숫자 데이터
        int number;
        // 문자열 데이터
        String message;

        /** 데이터 2개를 이용하여 초기화하는 생성자 */
        public SimpleData(int num, String msg) {
                number = num;
                message = msg;
        }
        /** 다른 Parcel 객체를 이용해 초기화하는 생성자 */
        public SimpleData(Parcel src) {
                number = src.readInt();
                message = src.readString();
        }
        /** 내부의 CREATOR 객체 생성 */
        @SuppressWarnings("unchecked")
        public static final Creator CREATOR = new Creator() {
                public SimpleData createFromParcel(Parcel in) {
                        return new SimpleData(in);
                }
                public SimpleData[] newArray(int size) {
                        return new SimpleData[size];
                }
        };
```

- 두 개의 멤버변수로 구성된 객체
- 이 데이터를 필요에 따라 Parcel 객체로 만드는 writeToParcel() 메서드 안을 보면 writeInt() , writeString() 메서드를 이용해 Parcel객체로 데이터를 쓴다
- 생성자 - Parcel객체를 파라미터로 받게 되는데 readInt(), readString() 메소드를 이용해 데이터를 읽어들임
- CREATOR 객체 – 상수로 정의되어 있는 객체, 새로운 객체가 만들어지는 코드가 들어가므로, new SimpleData(in) 와 같이 SimpleData 객체를 만드는 부분이 있다

⇒SimpleData 클래스 안에 Parcel객체의 데이터를 읽는 부분과 Parcel 객체로 쓰는 부분을 정의하게 됨

```
	public int describeContents() {
		return 0;
	}

	/**
	 * 데이터를 Parcel 객체로 쓰기
	 */
	public void writeToParcel(Parcel dest, int flags) {
		dest.writeInt(number);
		dest.writeString(message);
	}

	public int getNumber() {
		return number;
	}

	public void setNumber(int number) {
		this.number = number;
	}

	public String getMessage() {
		return message;
	}

	public void setMessage(String message) {
		this.message = message;
	}

}
```

```
public class AnotherActivity extends Activity {
    /** 부가 데이터를 위해 정의한 키 값        */
    public static final String KEY_SIMPLE_DATA = "data";

    TextView txtMsg;

   @Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.another);

       txtMsg = (TextView) findViewById(R.id.txtMsg);
       Button backBtn = (Button) findViewById(R.id.backButton);

      // 버튼을 눌렀을 때 메인 액티비티로 돌아갑니다.
      backBtn.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
              // 객체를 만듭니다.
              Intent resultIntent = new Intent();
              resultIntent.putExtra("name", "mike");

              // 응답을 전달하고 이 액티비티를 종료합니다.
              setResult(RESULT_OK, resultIntent);
              finish();
        }
       });

       // 전달된 인텐트를 처리합니다.
       processIntent();
   }
```

- 메인 액티비티로부터 전달된 인텐트 객체를 참조하기 위해 getIntent() 메서드가 사용
- 인텐트 객체 안의 번들 객체를 참조하기 위해 getExtras() 메서드도 사용됨
- 번들 객체 안에 SimpleData 객체가 들어 있으므로 getParcelable() 메서드를 이용해 객체를 참조한 후 화면의 텍스트뷰에 전달된 데이터를 보여줌
- 번들 객체에 데이터를 저장하기 위한 키 값은 메인 액티비티와 또 다른 액티비티 모두 동일한 상수로 정의되어 있다

```
    /** 전달된 인텐트 처리      */
    private void processIntent() {
        // 인텐트 안의 번들 객체를 참조합니다.
        Bundle bundle = getIntent().getExtras();

        // 번들 객체 안의 SimpleData 객체를 참조합니다.
        SimpleData data = (SimpleData)bundle.getParcelable(KEY_SIMPLE_DATA);

        // 텍스트뷰에 값을 보여줍니다.
        txtMsg.setText("Parcelable 객체로 전달된 값₩nNumber : " + data.getNumber() + "₩nMessage : " +
                data.getMessage());
    }

}
```

메인 액티비티 코드 만들기

```
...
public static final String KEY_SIMPLE_DATA = "data";
...
  launchBtn.setOnClickListener(new OnClickListener(){
   public void onClick(View v) {
    Intent intent = new Intent(getApplicationContext(), AnotherActivity.class);
    SimpleData data = new SimpleData(100, "Hello Android!" );
    intent.putExtra(KEY_SIMPLE_DATA, data );
    startActivity(intent);
    }
  });
```

1 SimpleData 객체 생성

2 인텐트에 부가 데이터로 넣기

SimpleData 클래스 정의

```
public class SimpleData implements Parcelable {
  int number;
  String message;
  public SimpleData(int num, String msg) {
    number = num;
    message = msg;
  }

  public SimpleData(Parcel src) {
    number = src.readInt();
    message = src.readString();
  }

  public static final Parcelable.Creator CREATOR = new Parcelable.Creator() {
     public SimpleData createFromParcel(Parcel in) {
       return new SimpleData (in);
     }
  public SimpleData[] newArray(int size) {
    return new SimpleData[size];
  }
};
```

Continued..

SimpleData 클래스 정의 (계속)

```
  public int describeContents() {
    return 0;
  }
  public void writeToParcel(Parcel dest, int flags) {
    dest.writeInt(number);
    dest.writeString(message);
  }
  public int getNumber() {
    return number;
  }
  public void setNumber(int number) {
    this.number = number;
  }
  public String getMessage() {
    return message;
  }
  public void setMessage(String message) {
    this.message = message;
  }
}
```

4 객체 쓰기

또 다른 액티비티 코드 만들기

```
public class AnotherActivity extends Activity {
 public static final String KEY_SIMPLE_DATA = "data";
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.another);

  TextView txtMsg = (TextView) findViewById(R.id.txtMsg);

  Bundle bundle = getIntent().getExtras();
  SimpleData data = (SimpleData)bundle.getParcelable(KEY_SIMPLE_DATA );
  txtMsg.setText("Number : " + data.getNumber() + "₩nMessage : " +
                 data.getMessage());
 }
}
```

1 인텐트의 번들 객체 참조

2 Parcelable 인터페이스 형태로 된 SimpleData 객체 참조

4. 수명주기

수명 주기

- 안드로이드에서는 **실행되는 애플리케이션의 상태를 시스템에서 직접 관리**함
- 대부분의 휴대단말용 OS에서 사용하는 방법
- 독립적인 애플리케이션이 시스템에 의해 관리되지 않을 경우 실행된 애플리케이션이 메모리를 과도하게 점유하거나 화면을 보여주는 권한을 과도하게 가지게 됨으로써 전화기의 원래 기능인 전화수신/발신 기능 자체를 사용하지 못하게 될 수도 있기 때문

- 액티비티는 처음 실행될 때 메모리에 만들어지는 과정부터 시작해 실행과 중지 그리고 메모리에서 해제되는 여러 과정을 상태정보로 가지고 있게 되고 이러한 상태정보는 시스템에서 관리하면서 **각각의 상태에 해당하는 메서드를 자동으로 호출**하게 됨
- 예) 액티비티에 기본으로 만들어져 있는 onCreate() 메서드는 액티비티가 만들어질 때 시스템에서 자동으로 호출하는 메서드임

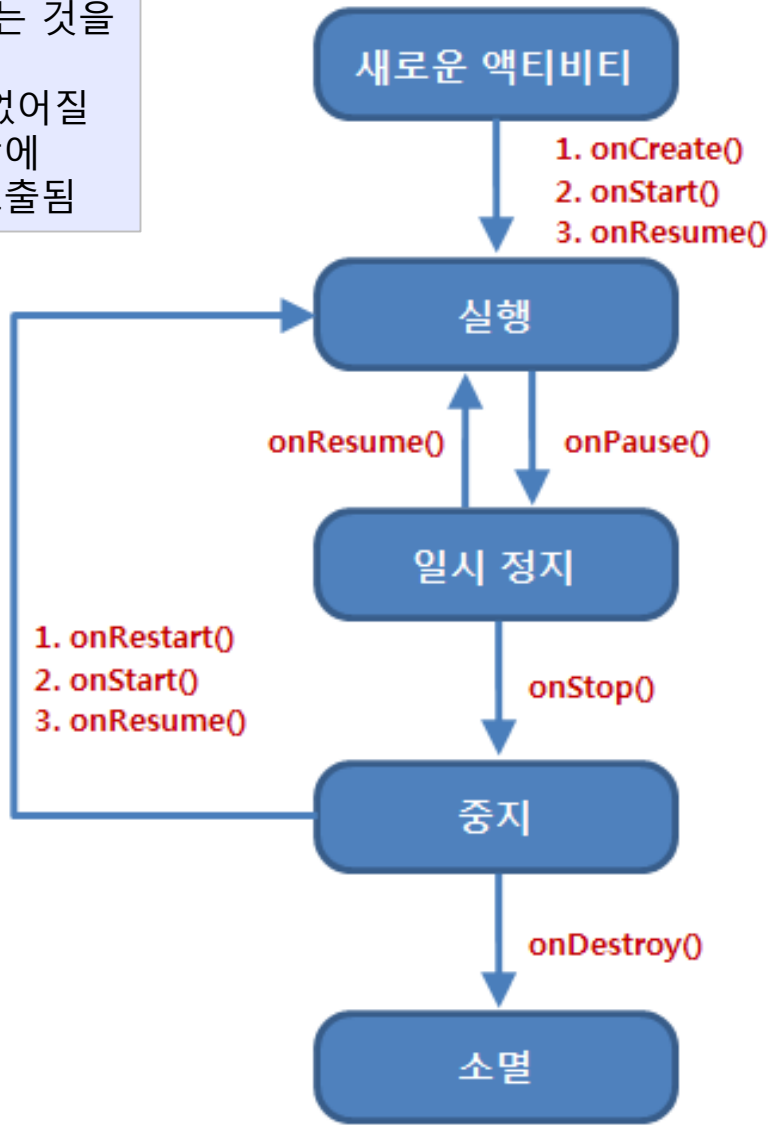
수명 주기

상 태	설 명
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 상태정보가 변화하는 것을 액티비티의 '수명주기'라 하며 액티비티가 처음 만들어진 후 없어질 때까지 상태가 변화하면서 각각에 해당하는 메서드가 자동으로 호출됨



새로운 액티비티가 만들어진다면 onCreate(), onStart(), onResume() 메서드가 차례대로 호출되며 그런 다음 화면에 보이게 됨

다른 액티비티가 상위에 오게 되면 onPause()메서드가 호출되면서 일시정지나 중지 상태로 변하게 됨

onStop() 메서드는 중지 상태로 변경될 때 자동으로 호출되는 메서드임

다시 실행될 때는 onResume() 메서드가 호출됨

액티비티가 메모리상에서 없어질 경우에는 onDestroy() 메서드가 호출됨

◀ 액티비티 수명주기

액티비티의 상태 메소드

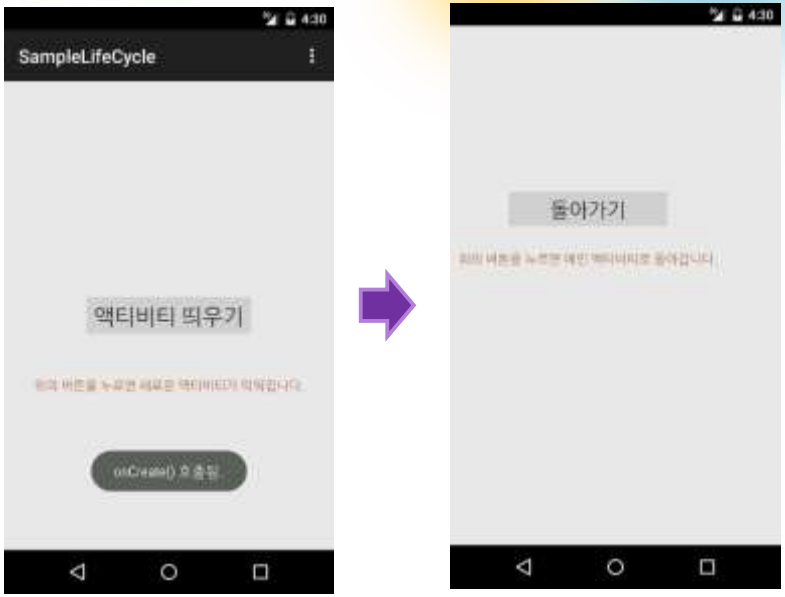
상태 메소드	설 명
onCreate()	- 액티비티가 처음에 만들어졌을 때 호출됨 - 화면에 보이는 뷰들의 일반적인 상태를 설정하는 부분 - 이전 상태가 저장되어 있는 경우에는 번들 객체를 참조하여 이전 상태 복원 가능 - 이 메소드 다음에는 항상 onStart() 메소드가 호출됨
onStart()	- 액티비티가 화면에 보이기 바로 전에 호출됨 - 액티비티가 화면 상에 보이면 이 메소드 다음에 onResume() 메소드가 호출됨 - 액티비티가 화면에서 가려지게 되면 이 메소드 다음에 onStop() 메소드가 호출됨
onResume()	- 액티비티가 사용자와 상호작용하기 바로 전에 호출됨
onRestart()	- 액티비티가 중지된 이후에 호출되는 메소드로 다시 시작되기 바로 전에 호출됨 - 이 메소드 다음에는 항상 onStart() 메소드가 호출됨
onPause()	- 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()	- 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()	- 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

메인 액티비티 코드 작성

-상태 메소드 별로 토스트 메시지 코드 추가

메인 액티비티 코드 만들기 (계속)

```
 @Override
protected void onDestroy() {
 super.onDestroy();
 Toast.makeText(getBaseContext(), "onDestroy ...", Toast.LENGTH_LONG).show();
}
@Override
protected void onPause() {
 super.onPause();
 saveCurrentState();
 Toast.makeText(getBaseContext(), "onPause ...", Toast.LENGTH_LONG).show();
}
@Override
protected void onRestart() {
 super.onRestart();
 Toast.makeText(getBaseContext(), "onRestart ...", Toast.LENGTH_LONG).show();
}
```

4 현재 상태 저장

Continued..

메인 액티비티 코드 만들기 (계속)

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

5 현재 상태 복원

Continued..

메인 액티비티 코드 만들기 (계속)

restoreFromSavedState() 메서드 – 설정 정보에 저장된 데이터를 가져와서 토스트 메시지로 보여줌
상태정보를 담고 있는 데이터를 저장하고 다시 복원하는 메서드는 onPause(), onResume() 메서드에 들어가야 액티비티가 화면상에서 사라지거나 또는 다시 화면으로 복원될 때 그 상태 그대로 사용자에게 보여줄 수 있다

```
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

saveCurrentState() 메서드 – 현재 입력되어 있는 데이터를 저장하는 메서드로써 문자열을 설정정보로 저장함
설정 정보는 데이터를 저장하는 가장 간단한 방법으로 SharedPreferences 객체를 getSharedPreperences() 메서드로 참조한 후 데이터를 저장함
SharedPreferences.Editor 객체는 데이터를 저장할 수 있도록 edit() 메서드를 제공하는데 edit()메서드를 호출한 후 putXXX() 메서드를 이용해 **데이터를 설정 정보에 추가**할 수 있다
데이터를 저장한 후에는 commit() 메서드를 호출해야 실제로 저장됨

수명주기 확인하기 실행 화면

액티비티 띄우기 버튼을 누르면 또 다른 액티비티가 띄워지면서 이에 따른 상태 메시지들도 표시됨
[돌아가기]버튼을 누르면 메인 액티비티가 다시 보이면서 저장되었던 데이터도 토스트 메시지로 표시됨

MainActivity

```
public class MainActivity extends AppCompatActivity {
    public static final int REQUEST_CODE_ANOTHER = 1001;
    public static final String PREF_ID = "Pref01";
    public static final int actMode = Activity.MODE_PRIVATE;

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

        Toast.makeText(this, "onCreate() 호출됨.", Toast.LENGTH_LONG).show();
    }

    public void onButton1Clicked(View v) {
        // 인텐트 객체를 만듭니다.
        Intent intent = new Intent(this, AnotherActivity.class);

        // 액티비티를 띄워주도록 startActivityForResult() 메소드를 호출합니다.
        startActivityForResult(intent, REQUEST_CODE_ANOTHER);
    }
    @Override
    protected void onDestroy() {
        Toast.makeText(this, "onDestroy() 호출됨.", Toast.LENGTH_LONG).show();

        super.onDestroy();
    }
```

MainActivity

```
@Override
protected void onPause() {
    Toast.makeText(this, "onPause() 호출됨.", Toast.LENGTH_LONG).show();
    saveCurrentState();
    super.onPause();
}

@Override
protected void onRestart() {
    Toast.makeText(this, "onRestart() 호출됨.", Toast.LENGTH_LONG).show();
    super.onRestart();
}

@Override
protected void onResume() {
    Toast.makeText(this, "onResume() 호출됨.", Toast.LENGTH_LONG).show();
    restoreFromSavedState();
    super.onResume();
}

@Override
protected void onStart() {
    Toast.makeText(this, "onStart() 호출됨.", Toast.LENGTH_LONG).show();
    super.onStart();
}
```

MainActivity

```
@Override
protected void onStop() {
    Toast.makeText(this, "onStop() 호출됨.", Toast.LENGTH_LONG).show();
    super.onStop();
}

/** 새로운 액티비티에서 돌아올 때 자동 호출되는 메소드  */
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE_ANOTHER) {
        Toast toast = Toast.makeText(this, "onActivityResult() 메소드가 호출됨. 요청코드 : " + requestCode + ", 결과코드 : " +
  resultCode, Toast.LENGTH_LONG);
        toast.show();
    }
}

protected void restoreFromSavedState() {
    SharedPreferences myPrefs = getSharedPreferences(PREF_ID, actMode);
    if ((myPrefs != null) && (myPrefs.contains("txtMsg")) ) {
        String myData = myPrefs.getString("txtMsg", "");
        Toast.makeText(this, "Restored : " + myData, Toast.LENGTH_SHORT).show();
    }
}
```

MainActivity

```
  protected void saveCurrentState() {
      SharedPreferences myPrefs = getSharedPreferences(PREF_ID, actMode);
      SharedPreferences.Editor myEditor = myPrefs.edit();
      myEditor.putString( "txtMsg", "My name is mike." );
      myEditor.commit();
    }

    protected void clearMyPrefs() {
      SharedPreferences myPrefs = getSharedPreferences(PREF_ID, actMode);
      SharedPreferences.Editor myEditor = myPrefs.edit();
      myEditor.clear();
      myEditor.commit();
    }

}
```

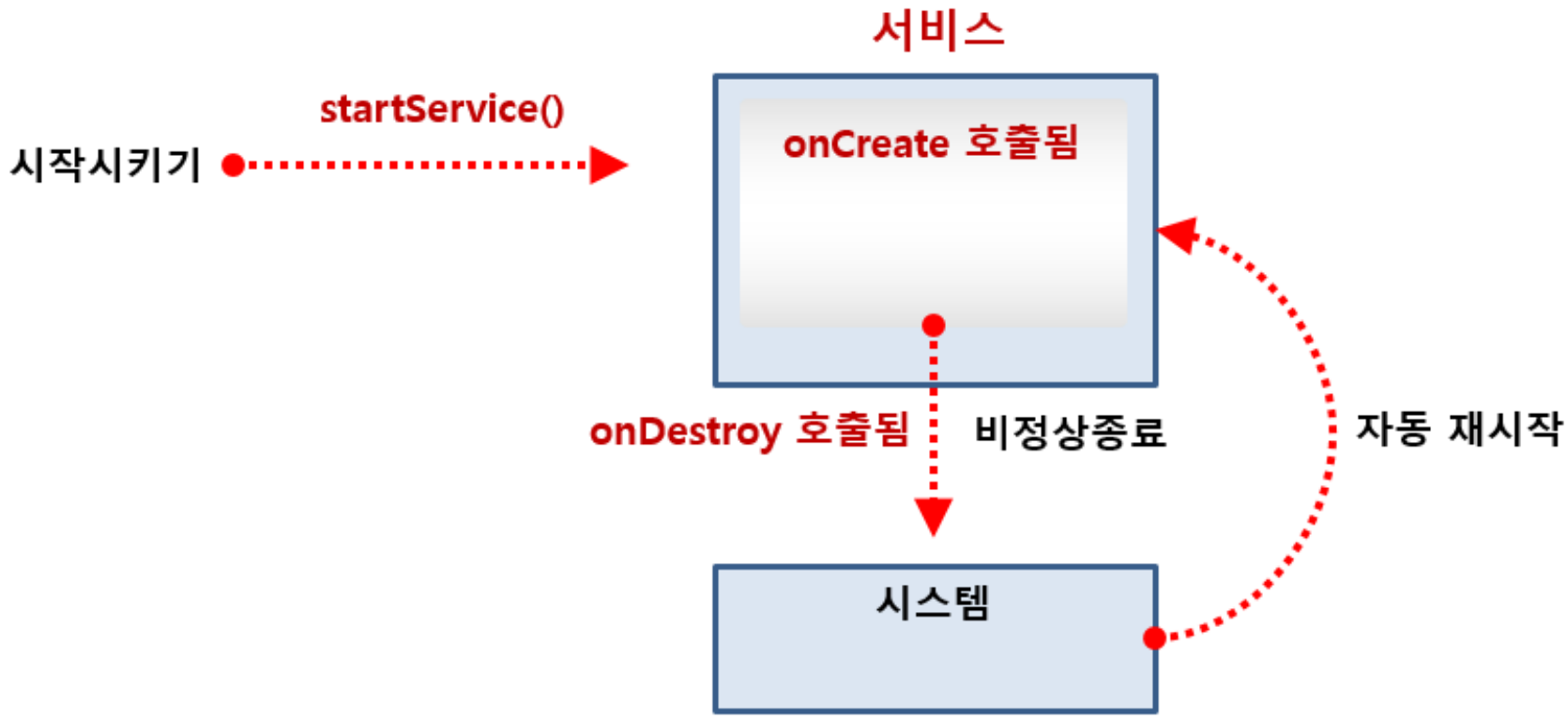
```
public class AnotherActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.another);

        Button backBtn = (Button) findViewById(R.id.backBtn);
                // 버튼을 눌렀을 때 메인 액티비티로 돌아갑니다.
                backBtn.setOnClickListener(new OnClickListener() {
                        public void onClick(View v) {
                                // 객체를 만듭니다.
                                Intent resultIntent = new Intent();
                                resultIntent.putExtra("name", "mike");

                                // 응답을 전달하고 이 액티비티를 종료합니다.
                                setResult(RESULT_OK, resultIntent);
                                finish();
                        }
                });
      Toast.makeText(this, "새로운 액티비티의 onCreate() 호출됨.", Toast.LENGTH_LONG).show();
    }
    @Override
      protected void onDestroy() {
                Toast.makeText(this, "새로운 액티비티의 onDestroy() 호출됨.", Toast.LENGTH_LONG).show();
                super.onDestroy();
      }
```

```
    @Override
    protected void onPause() {
        Toast.makeText(this, "새로운 액티비티의 onPause() 호출됨.", Toast.LENGTH_LONG).show();
        super.onPause();
    }
    @Override
    protected void onRestart() {
        Toast.makeText(this, "새로운 액티비티의 onRestart() 호출됨.", Toast.LENGTH_LONG).show();
        super.onRestart();
    }
    @Override
    protected void onResume() {
        Toast.makeText(this, "새로운 액티비티의 onResume() 호출됨.", Toast.LENGTH_LONG).show();
        super.onResume();
    }
    @Override
    protected void onStart() {
        Toast.makeText(this, "새로운 액티비티의 onStart() 호출됨.", Toast.LENGTH_LONG).show();
        super.onStart();
    }
    @Override
    protected void onStop() {
        Toast.makeText(this, "새로운 액티비티의 onStop() 호출됨.", Toast.LENGTH_LONG).show();
        super.onStop();
    }
}
```

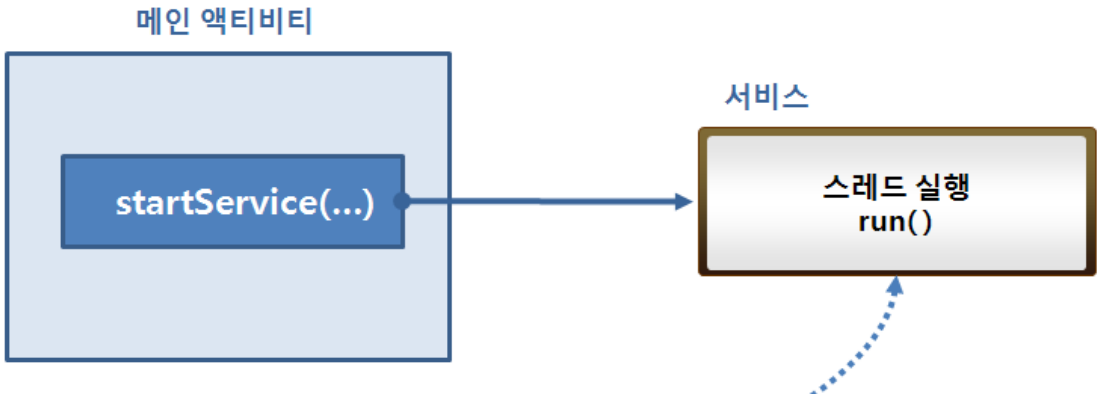
5. 서비스

자동으로 재시작되는 서비스

- 서비스는 화면이 없는 상태에서 백그라운드로 실행됨
- 서비스는 프로세스가 종료되어도 시스템에서 자동으로 재시작함



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(...)

• 안드로이드에서 서비스 – 백그라운드에서 실행되는 프로세스
• 액티비티와 다른점 – 화면이 없다는 것
• 화면이 없다는 것을 제외하면 사실상 애플리케이션의 구성요소로서 액티비티처럼 동작한다고 생각할 수 있다
• 서비스도 애플리케이션의 구성요소이므로 새로 만든 후에는 항상 매니페스트에 등록해야 하며, 메인 액티비티에서 서비스를 시작하고 싶은 경우에는 startService() 메서드를 이용해 서비스를 시작시킬 수 있다

- 서비스는 다른 구성 요소들처럼 메인 스레드에서 동작

따라서 CPU를 많이 쓰거나 대기 상태(blocking)를 필요로 하는 작업들은 **스레드를 새로** 만들어 주어야 함

서비스 클래스 정의

```
public class MyService extends Service {
…
  @Override
  public void onCreate() {
    super.onCreate();
    Log.d(TAG, "onCreate() 호출됨.");
  }

  @Override
  public int onStartCommand(Intent intent, int flags, int startId) {
    Log.d(TAG, "onStartCommand() 호출됨.");

    if (intent == null) {
      return Service.START_STICKY;
    } else {
      processCommand(intent);
    }

    return super.onStartCommand(intent, flags, startId);
  }
…
```

Continued..

서비스 클래스 정의

```
...

  private void processCommand(Intent intent) {
    String command = intent.getStringExtra("command");
    String name = intent.getStringExtra("name");
    Log.d(TAG, "command : " + command + ", name : " + name);

    for (int i = 0; i < 5; i++) {
      try {
        Thread.sleep(1000);
      } catch(Exception e) {};

      Log.d(TAG, "Waiting " + i + " seconds.");
    }
  }

...
```

Continued..

메인 액티비티 코드 만들기

참조파일 SampleService>/app/java/org.techtown.service/MainActivity.java

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

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

    editText = findViewById(R.id.editText);

    Button button = findViewById(R.id.button);
    button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        String name = editText.getText().toString();

        Intent intent = new Intent(getApplicationContext(), MyService.class);
        intent.putExtra("command", "show");
        intent.putExtra("name", name);

        startService(intent);
      }
    });
  }
}
```

❶ 인텐트 객체 만들고 부가 데이터 넣기

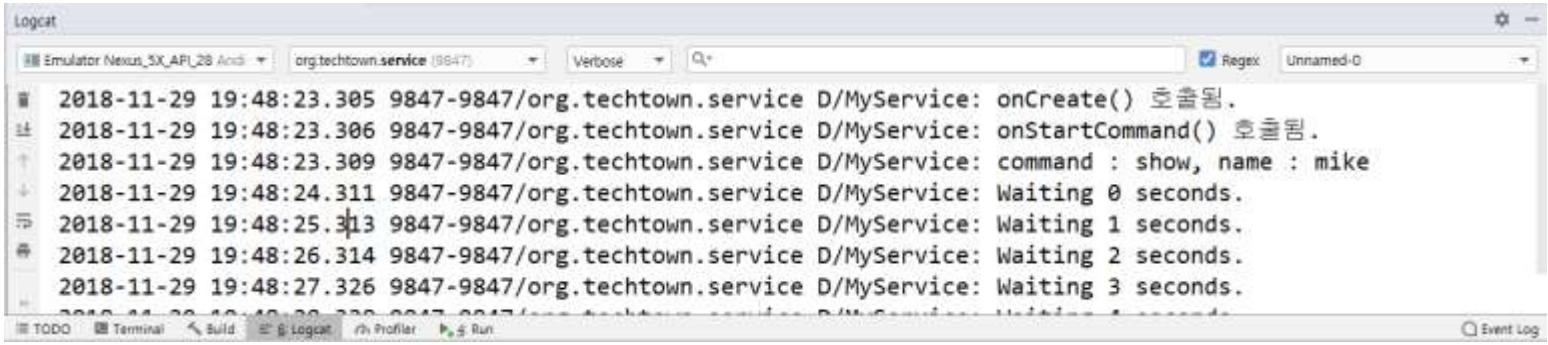
❷ 서비스 시작하기

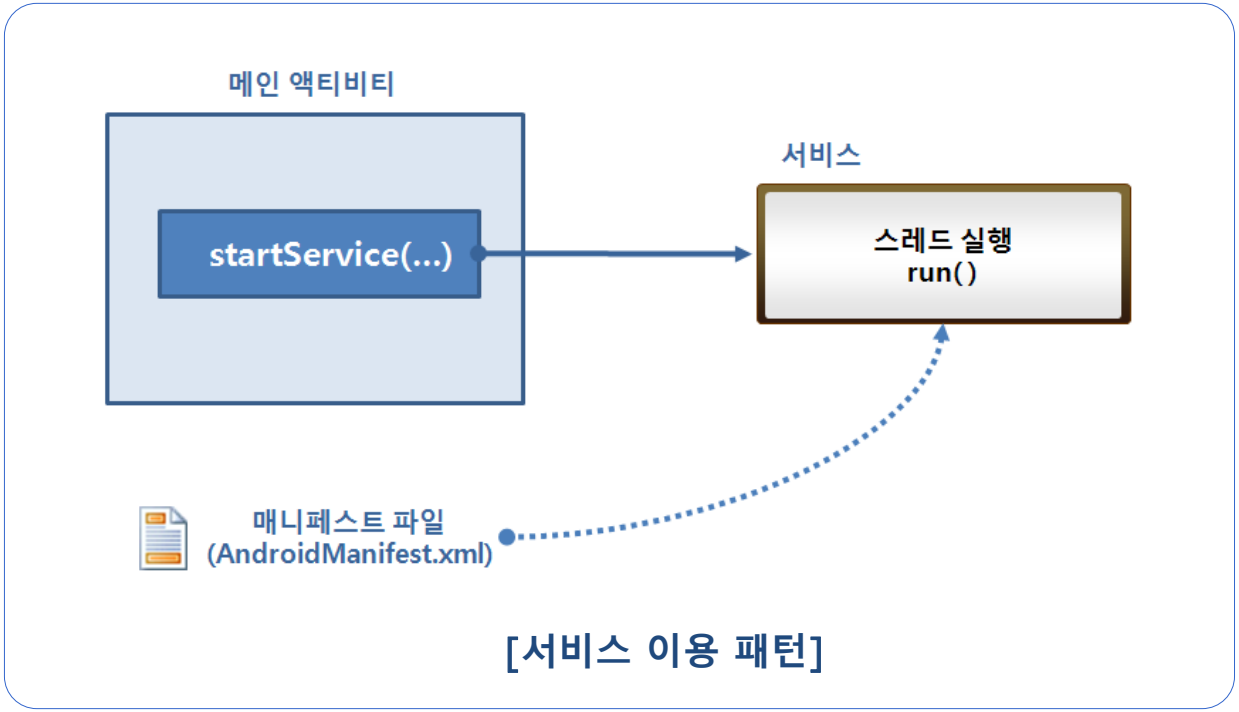
서비스는 매니페스트에 자동 추가됨

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

</application>
...
```

1 서비스 등록

서비스 실행하여 로그 확인





서비스에서 화면 띄우기

- 서비스에서 액티비티를 띄울 수 있음
- 플래그를 이용해 한 번 만들어진 액티비티를 그대로 띄움

서비스는 화면이 없는 상태에서 화면이 있는 액티비티를 띄워주게 되므로 새로운 태스트(Task)를 만들어 띄워주어야 함

태스크는 일련의 화면이 처리되는 과정을 묶어놓은 것으로, 개발자가 직접 만든 앱에서 다른 앱의 화면을 띄워줄 때 서로 다른 프로세스에서 동작하는 화면을 서로 묶어줄 필요가 있어 만든 것임

이 태스크는 화면이 없이 동작하는 서비스나 브로드캐스트 수신자에서 화면을 띄울 때도 사용됨

액티비티 화면을 띄워줄 때 FLAG_ACTIVITY_SINGLE_TOP 플래그와 FLAG_ACTIVITY_CLEAR_TOP 플래그를 같이 주면 해당 액티비티가 메모리에 만들어져 있을 때는 새로 만들지 않고 그대로 보여줌

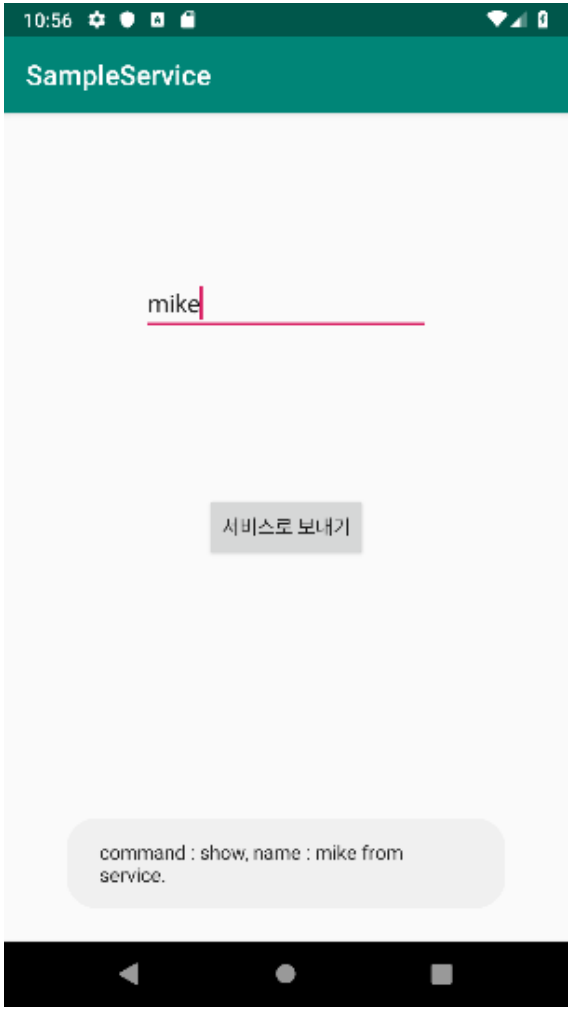
```
private void processCommand(Intent intent) {
  ...

  Intent showIntent = new Intent(getApplicationContext(), MainActivity.class);
  showIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
  Intent.FLAG_ACTIVITY_SINGLE_TOP |
  Intent.FLAG_ACTIVITY_CLEAR_TOP);
  showIntent.putExtra("command", "show");
  showIntent.putExtra("name", name + " from service.");
  startActivity(showIntent);
}
```

액티비티에서 인텐트 받아 처리하기

- 액티비티가 이미 메모리에 만들어져 있는 경우 onNewIntent 메소드 호출됨

```
@Override
protected void onNewIntent(Intent intent) {
  processIntent(intent);
  super.onNewIntent(intent);
}

private void processIntent(Intent intent) {
  if (intent != null) {
    String command = intent.getStringExtra("command");
    String name = intent.getStringExtra("name");

    Toast.makeText(this, "command : " + command + ", name : " + name,
                   Toast.LENGTH_LONG).show();
  }
}
```

서비스에서 띄운 화면


10:56
SampleService
mike
서비스로 보내기
command : show, name : mike from service.

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="서비스로 보내기"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp"
        android:ems="10"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

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

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

        editText = findViewById(R.id.editText);

        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String name = editText.getText().toString();

                Intent intent = new Intent(getApplicationContext(), MyService.class);
                intent.putExtra("command", "show");
                intent.putExtra("name", name);
                startService(intent);
            }
        });

        Intent passedIntent = getIntent();
        processIntent(passedIntent);
    }
```

```
    @Override
    protected void onNewIntent(Intent intent) {
        processIntent(intent);

        super.onNewIntent(intent);
    }

    private void processIntent(Intent intent) {
        if (intent != null) {
            String command = intent.getStringExtra("command");
            String name = intent.getStringExtra("name");
            Toast.makeText(this, "command : " + command + ", name : " + name,
                    Toast.LENGTH_LONG).show();
        }
    }

}
```

```
public class MyService extends Service {
    private static final String TAG = "MyService";

    @Override
    public void onCreate() {
        super.onCreate();

        Log.d(TAG, "onCreate() 호출됨.");
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.d(TAG, "onStartCommand() 호출됨.");

        if (intent == null) {
            return Service.START_STICKY;
        } else {
            processCommand(intent);
        }

        return super.onStartCommand(intent, flags, startId);
    }

    private void processCommand(Intent intent) {
        String command = intent.getStringExtra("command");
        String name = intent.getStringExtra("name");
```

```
        Log.d(TAG, "command : " + command + ", name : " + name);

        for (int i = 0; i < 5; i++) {
            try {
                Thread.sleep(1000);
            } catch (Exception e) {
            }

            Log.d(TAG, "Waiting " + i + " seconds.");
        }

        Intent showIntent = new Intent(getApplicationContext(), MainActivity.class);
        showIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
                Intent.FLAG_ACTIVITY_SINGLE_TOP |
                Intent.FLAG_ACTIVITY_CLEAR_TOP);
        showIntent.putExtra("command", "show");
        showIntent.putExtra("name", name + " from service.");
        startActivity(showIntent);
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

서비스는 화면이 없는 상태에서 화면이 있는 액티비티를 띄워주게 되므로 새로운 태스트(Task)를 만들어 띄워주어야 함

태스크는 일련의 화면이 처리되는 과정을 묶어놓은 것으로, 개발자가 직접 만든 앱에서 다른 앱의 화면을 띄워줄 때 서로 다른 프로세스에서 동작하는 화면을 서로 묶어줄 필요가 있어 만든 것임
이 태스크는 화면이 없이 동작하는 서비스나 브로드캐스트 수신자에서 화면을 띄울 때도 사용됨

액티비티 화면을 띄워줄 때 FLAG_ACTIVITY_SINGLE_TOP 플래그와 FLAG_ACTIVITY_CLEAR_TOP 플래그를 같이 주면 해당 액티비티가 메모리에 만들어져 있을 때는 새로 만들지 않고 그대로 보여줌

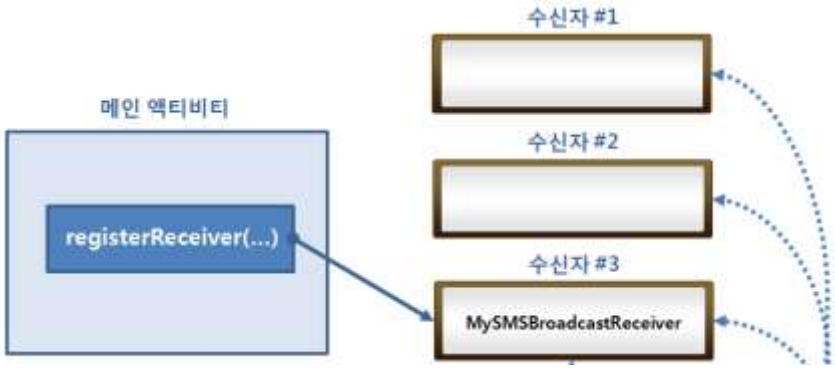
6. 브로드캐스트 수신자

브로드캐스트 수신자

브로드캐스팅이란

- 메시지를 여러 객체에게 전달하는 방법
- 일반적으로 채팅 애플리케이션을 구성할 때 볼 수 있는 개념
- 일대일 채팅을 할 때는 메시지를 두 사람 간에 주고 받지만 여러 사람에게 한꺼번에 전달하고 싶은 글이 있을 경우에는 브로드캐스팅 기능을 사용하는 것과 유사함
- 안드로이드에서는 여러 객체에게 메시지를 전달하고 싶은 경우에 브로드캐스팅을 사용함
- 브로드캐스팅 메시지는 브로드캐스트 수신자(Broadcast Receiver)라는 애플리케이션 구성요소를 이용해 받을 수 있다
- 즉, 어떤 메시지를 받고 싶다고 등록하면 그 메시지가 이 애플리케이션에도 전달되는 방식임
- 수신하고 싶은 메시지가 있다면 그 메시지는 인텐트 필터를 이용해 등록함
- 즉, 브로드캐스트 메시지도 인텐트에 정보를 넣어 전달하는 방식을 사용하므로 이 인텐트를 받고 싶다면 인텐트 필터를 이용해 시스템에 알려주어야 함

브로드캐스트 수신자



- 애플리케이션이 글로벌 이벤트(global event)를 받아서 처리하려면 브로드캐스트 수신자로 등록
- **글로벌 이벤트**란 "전화가 왔습니다.", "문자 메시지가 도착했습니다."와 같이 안드로이드 시스템 전체에 보내지는 이벤트
- 브로드캐스트 수신자는 인텐트필터를 포함하며, 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
  - **브로드캐스트**를 이용해서 처리하면 **백그라운드(background)**로 동작하므로 사용자가 모름
  - 인텐트를 받으면 onReceive() 메소드가 자동으로 호출됨

- **브로드캐스트의 구분**

  브로드캐스트는 크게 두 가지 클래스로 구분됨

  - [1] **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

  - [2] **순차 브로드캐스트** (sendOrderedBroadcast() 메소드로 호출)

    한 번에 하나의 수신자에만 전달되므로 **순서대로 실행**됨. 중간에 취소하면 그 다음 수신자는 받지 못함. 수신자가 실행되는 순서는 인텐트 필터의 속성으로 정할 수 있음

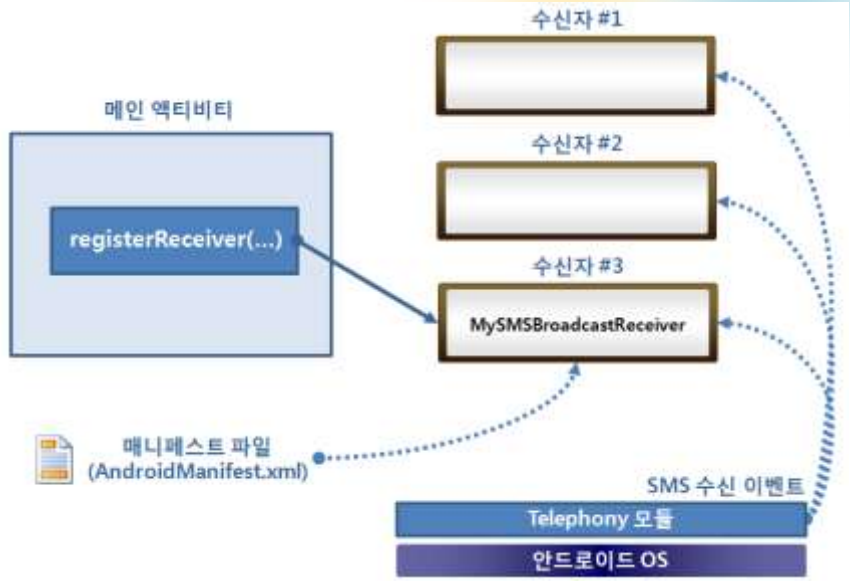
    순서가 같으면 임의로 실행됨.

브로드캐스트 수신자 예제

브로드캐스트 수신자 예제

-브로드캐스트 수신자로 SMS 수신 확인하기

-브로드캐스트 수신자 정의



브로드캐스트 수신자 정의

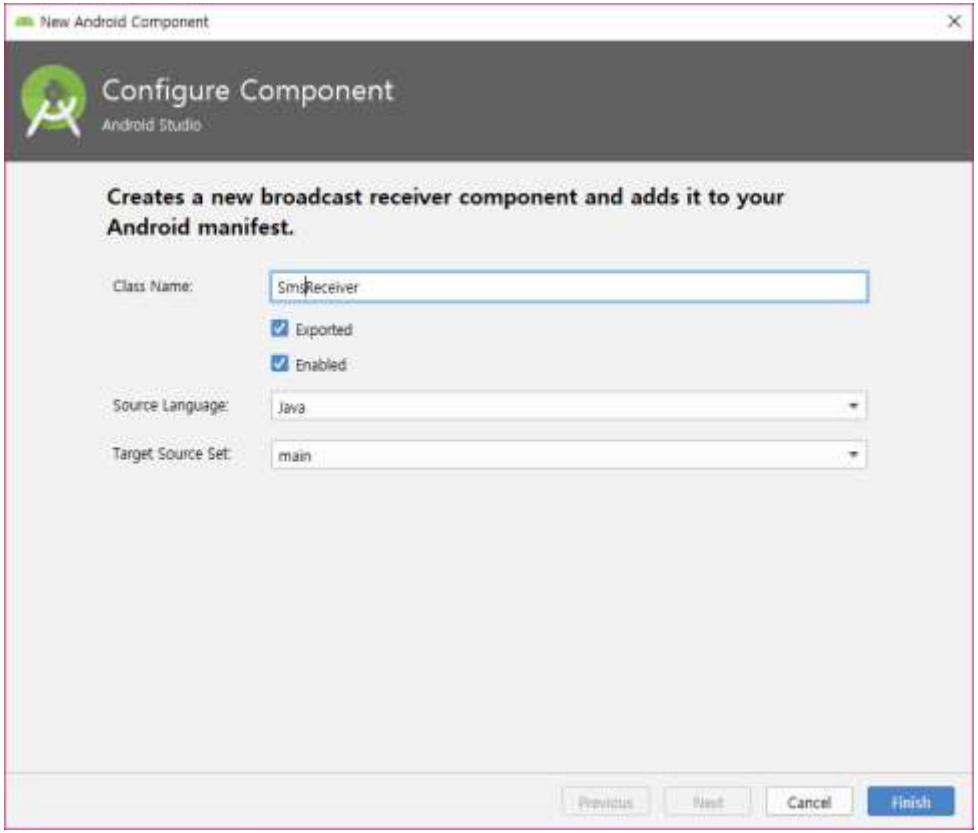
-일정 시간간격으로 메시지를 보여 주는 서비스 클래스 정의

매니페스트에 추가

-새로운 브로드캐스트 수신자를 매니페스트에 추가

새로운 브로드캐스트 수신자 추가

- SampleReceiver 프로젝트 생성
- New → Other → Broadcast Receiver 메뉴로 브로드캐스트 수신자 추가



매니페스트에 추가

```
<application
  ...

  <receiver
    android:name=".SmsReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
            <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
  </receiver>
</application>
```

브로드캐스트 수신자 클래스 정의

```
public class SmsReceiver extends BroadcastReceiver {
  public static final String TAG = "SmsReceiver";

  @Override
  public void onReceive(Context context, Intent intent) {
    Log.i(TAG, "onReceive() 메소드 호출됨.");

    Bundle bundle = intent.getExtras();
    SmsMessage[] messages = parseSmsMessage(bundle);
    if (messages != null && messages.length > 0) {
      String sender = messages[0].getOriginatingAddress();
      Log.i(TAG, "SMS sender : " + sender);
      String contents = messages[0].getMessageBody().toString();
      Log.i(TAG, "SMS contents : " + contents);
      Date receivedDate = new Date(messages[0].getTimestampMillis());
      Log.i(TAG, "SMS received date : " + receivedDate.toString());
    }

  }
...
```

브로드캐스트 메시지 수신 시 자동 호출됨

브로드캐스트 수신자 클래스 정의

```
...
private SmsMessage[] parseSmsMessage(Bundle bundle) {
  Object[] objs = (Object[]) bundle.get("pdus");
  SmsMessage[] messages = new SmsMessage[objs.length];

  int smsCount = objs.length;
  for (int i = 0; i < smsCount; i++) {
    // PDU 포맷으로 되어 있는 메시지를 복원합니다.
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) { // API 23 이상
      String format = bundle.getString("format");
      messages[i] = SmsMessage.createFromPdu((byte[]) objs[i], format);
    } else {
      messages[i] = SmsMessage.createFromPdu((byte[]) objs[i]);
    }
  }

  return messages;
}
...
```

Build.VERSION.**SDK_INT** - 단말의 **OS**버전을 확인할 때 사용

Build.VERSION_CODES.M 에는 안드로이드 OS버전별로 상수가 정의되어 있다
OS가 마시멜로(첫글자 M)버전과 같거나 그 이후 버전일 때 실행된다는 의미

권한 추가

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.androidtown.samplereceiver">

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

...
```

Sync Now 버튼 클릭


```
allprojects {
    repositories{
        maven {url 'https://jitpack.io'}
    }
}


dependencies {
    ...
    implementation 'com.github.pedroSG94:AutoPermissions:1.0.3'
}
```

위험 권한 부여 코드 추가

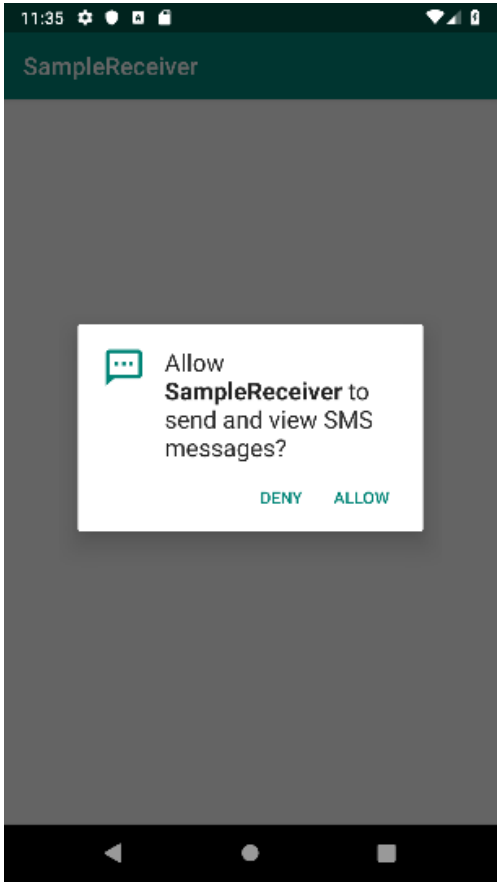
참조파일 SampleReceiver>/app/java/org.techtown.receiver/MainActivity.java

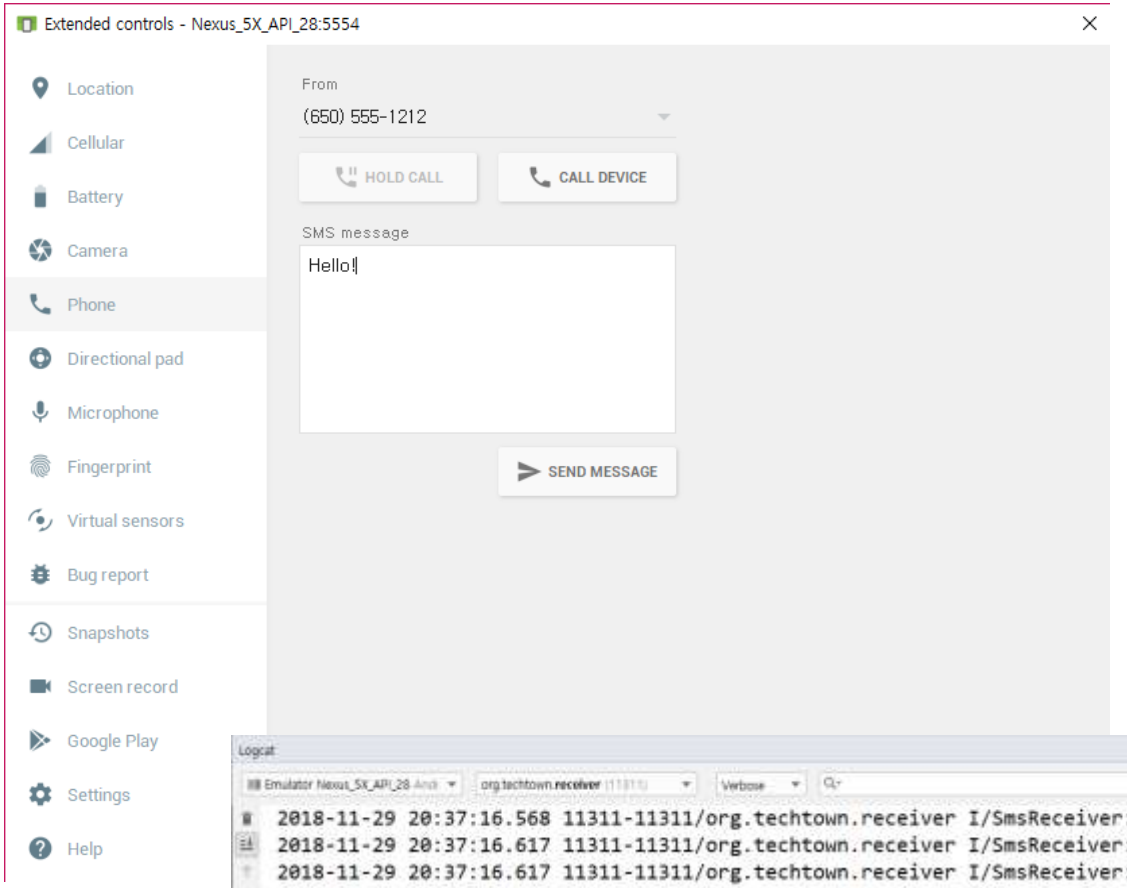
```
public class MainActivity extends AppCompatActivity
                    implements AutoPermissionsListener {

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

    AutoPermissions.Companion.loadAllPermissions(this, 101);
  }

  @Override  public void onRequestPermissionsResult(int requestCode, String permissions[],
                                      int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    AutoPermissions.Companion.parsePermissions(this, requestCode, permissions, this);
  }

  @Override
  public void onDenied(int requestCode, @NotNull String[] permissions) {
    Toast.makeText(this, "permissions denied : " + permissions.length,
                Toast.LENGTH_LONG).show();
  }

  @Override  public void onGranted(int requestCode, @NotNull String[] permissions) {
    Toast.makeText(this, "permissions granted : " + permissions.length,
                Toast.LENGTH_LONG).show();
  }
}
```

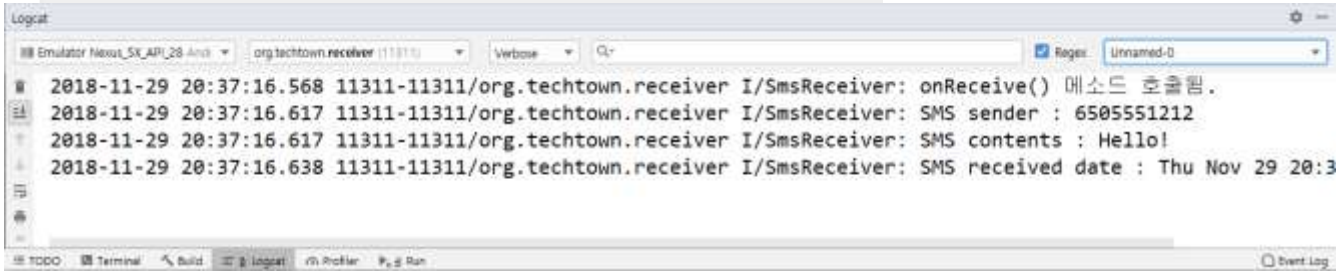
❶ MainActivity가 인터페이스 구현하도록 하기

❷ 모든 위험 권한을 자동 부여하도록 하는 메서드 호출하기

브로드캐스트 수신자 실행 화면

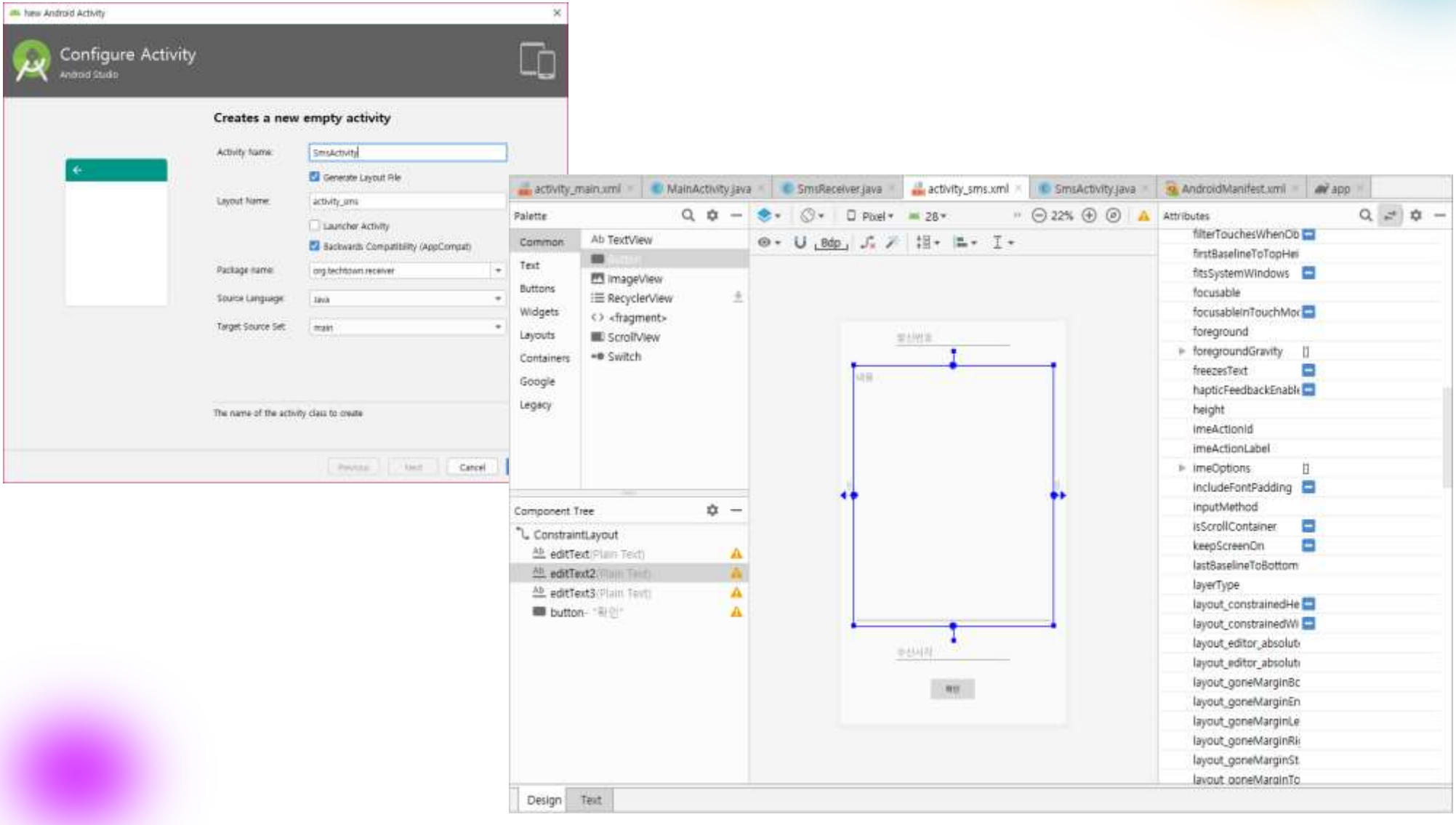






[Extended Controls 에서 에뮬레이터로 SMS를 보내고 로그가 출력된 화면]

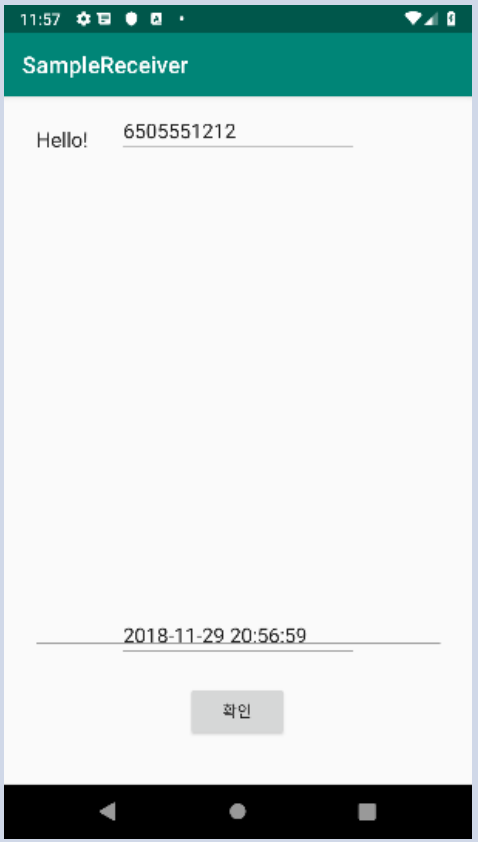
SMS 문자를 보여줄 새로운 화면 추가



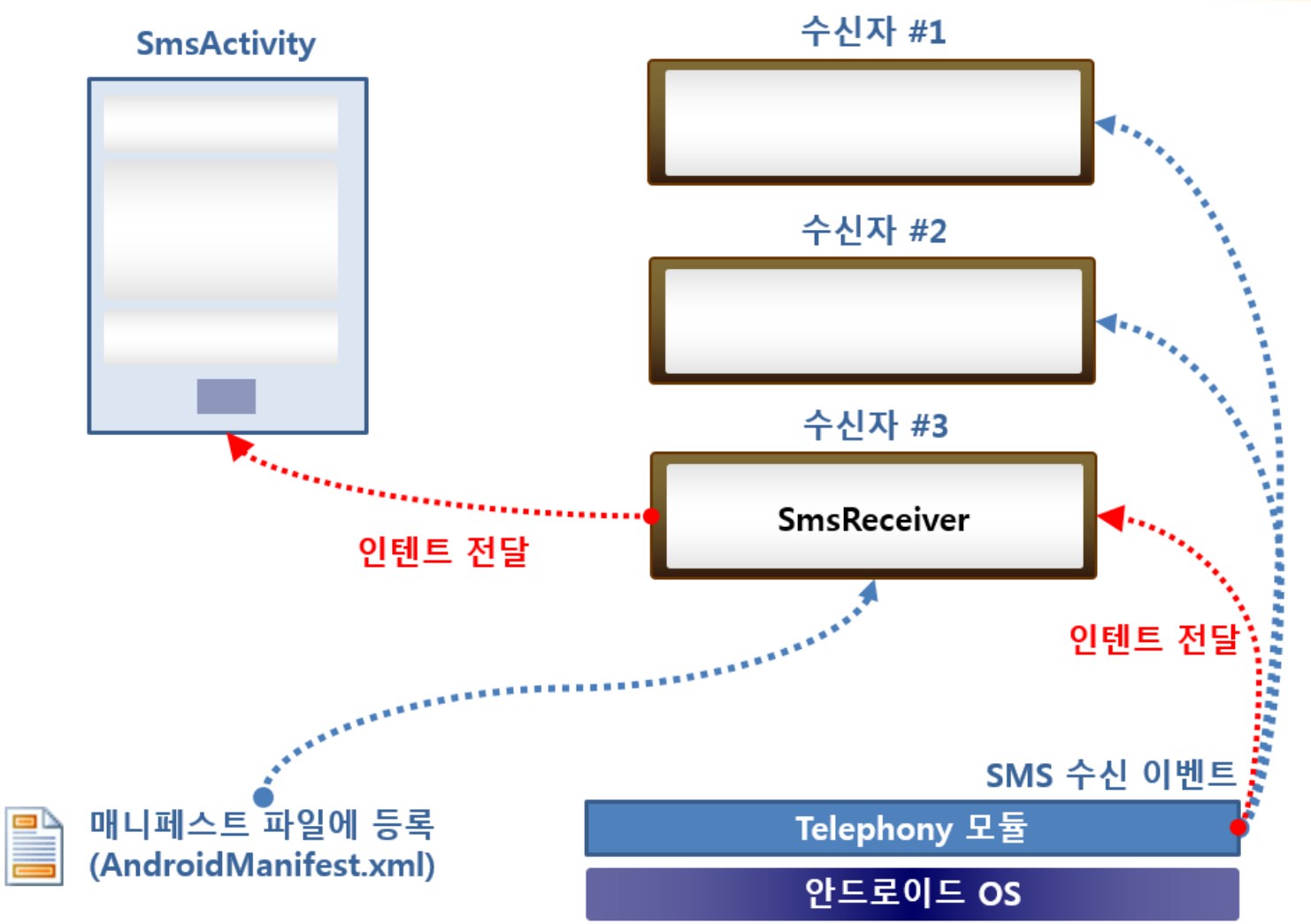
SMS 문자를 보여줄 새로운 화면 추가

```
...
@Override
protected void onNewIntent(Intent intent) {
  processIntent(intent);

  super.onNewIntent(intent);
}

private void processIntent(Intent intent) {
  if (intent != null) {
    String sender = intent.getStringExtra("sender");
    String contents = intent.getStringExtra("contents");
    String receivedDate = intent.getStringExtra("receivedDate");

    editText.setText(sender);
    editText2.setText(contents);
    editText3.setText(receivedDate);
  }
}
...
```

```
...
private void sendToActivity(Context context, String sender, String contents, Date receivedDate) {
  // 메시지를 보여줄 액티비티를 띄워줍니다.
  Intent myIntent = new Intent(context, SmsActivity.class);

  // 플래그를 이용합니다.
  myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK|
  Intent.FLAG_ACTIVITY_SINGLE_TOP|Intent.FLAG_ACTIVITY_CLEAR_TOP);

  myIntent.putExtra("sender", sender);
  myIntent.putExtra("contents", contents);
  myIntent.putExtra("receivedDate", format.format(receivedDate));

  context.startActivity(myIntent);
}
...
```



브로드캐스트 수신자 동작 방식



activity_main.xml

```
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

activity_sms.xml

```
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SmsActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:ems="10"
        android:hint="발신번호"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/editText2"
        android:layout_width="363dp"
        android:layout_height="472dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
```

```
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:ems="10"
        android:gravity="top|left"
        android:hint="내용"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toTopOf="@+id/editText3"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText" />
    <EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:ems="10"
        android:hint="수신시각"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="40dp"
        android:text="확인"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```

```
public class MainActivity extends AppCompatActivity implements AutoPermissionsListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        AutoPermissions.Companion.loadAllPermissions(this, 101);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);

        AutoPermissions.Companion.parsePermissions(this, requestCode, permissions, this);
    }

    @Override
    public void onDenied(int requestCode, @NotNull String[] permissions) {
        Toast.makeText(this, "permissions denied : " + permissions.length, Toast.LENGTH_LONG).show();
    }

    @Override
    public void onGranted(int requestCode, @NotNull String[] permissions) {
        Toast.makeText(this, "permissions granted : " + permissions.length, Toast.LENGTH_LONG).show();
    }
}
```

위험권한을 자동으로 부여하는 코드

@NotNull 에러시 dependencies 추가

implementation 'org.jetbrains:annotations:12.0'

```
public class SmsActivity extends AppCompatActivity {
    EditText editText;
    EditText editText2;
    EditText editText3;

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

        editText = findViewById(R.id.editText);
        editText2 = findViewById(R.id.editText2);
        editText3 = findViewById(R.id.editText3);

        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });

        Intent passedIntent = getIntent();
        processIntent(passedIntent);
    }
```

```
    @Override
    protected void onNewIntent(Intent intent) {
        processIntent(intent);

        super.onNewIntent(intent);
    }

    private void processIntent(Intent intent) {
        if (intent != null) {
            String sender = intent.getStringExtra("sender");
            String contents = intent.getStringExtra("contents");
            String receivedDate = intent.getStringExtra("receivedDate");

            editText.setText(sender);
            editText2.setText(contents);
            editText3.setText(receivedDate);
        }
    }

}
```

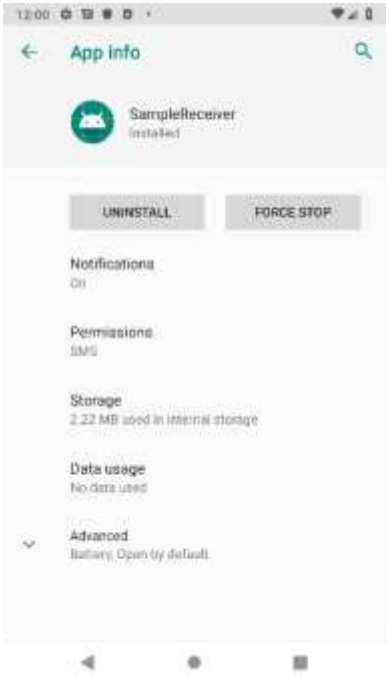
```
public class SmsReceiver extends BroadcastReceiver {
    public static final String TAG = "SmsReceiver";

    public SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i(TAG, "onReceive() 메소드 호출됨.");

        Bundle bundle = intent.getExtras();
        SmsMessage[] messages = parseSmsMessage(bundle);
        if (messages != null && messages.length > 0) {
            String sender = messages[0].getOriginatingAddress();
            Log.i(TAG, "SMS sender : " + sender);

            String contents = messages[0].getMessageBody().toString();
            Log.i(TAG, "SMS contents : " + contents);

            Date receivedDate = new Date(messages[0].getTimestampMillis());
            Log.i(TAG, "SMS received date : " + receivedDate.toString());

            sendToActivity(context, sender, contents, receivedDate);
        }
    }
```

```
    private SmsMessage[] parseSmsMessage(Bundle bundle) {
        Object[] objs = (Object[]) bundle.get("pdus");
        SmsMessage[] messages = new SmsMessage[objs.length];

        int smsCount = objs.length;
        for (int i = 0; i < smsCount; i++) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                String format = bundle.getString("format");
                messages[i] = SmsMessage.createFromPdu((byte[]) objs[i], format);
            } else {
                messages[i] = SmsMessage.createFromPdu((byte[]) objs[i]);
            }
        }
        return messages;
    }
    private void sendToActivity(Context context, String sender, String contents, Date receivedDate) {
        Intent myIntent = new Intent(context, SmsActivity.class);
        myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
                Intent.FLAG_ACTIVITY_SINGLE_TOP | Intent.FLAG_ACTIVITY_CLEAR_TOP);
        myIntent.putExtra("sender", sender);
        myIntent.putExtra("contents", contents);
        myIntent.putExtra("receivedDate", format.format(receivedDate));
        context.startActivity(myIntent);
    }
}
```

onReceive() 메서드 – 새로운 SMS 메서드가 도착하면 자동으로 호출
SMS메세지를 추출한 후, SMS 액티비티를 띄워줌
이때 서비스는 화면이 없는 상태에서 화면이 있는 액티비티를 띄워주게 되므로 새로운 태스트(Task)를 만들어 띄워주어야 함
태스크는 일련의 화면이 처리되는 과정을 묶어놓은 것으로, 개발자가 직접 만든 앱에서 다른 앱의 화면을 띄워줄 때 서로 다른 프로세스에서 동작하는 화면을 서로 묶어줄 필요가 있어 만든 것임
이 태스크는 화면이 없이 동작하는 서비스나 브로드캐스트 수신자에서 화면을 띄울 때도 사용됨
SMS 액티비티 화면을 띄워줄 때 FLAG_ACTIVITY_SINGLE_TOP 플래그와 FLAG_ACTIVITY_CLEAR_TOP 플래그를 같이 주면 SMS 액티비티가 메모리에 만들어져 있을 때는 새로 만들지 않고 그대로 보여줌

화면을 수정해도 이전 화면이 계속 보여요!

- **프로젝트를 복사하여 새로운 프로젝트를 만든 경우**
  - SMS를 수신하는 동일한 앱이 설치되어 있다면 SMS 수신 시 그 화면이 보여질 수 있음
  - 따라서, 이전에 설치한 앱을 삭제한 후 새로운 앱을 실행해야 함
  - 단말의 설정에서 앱을 삭제할 수 있음



7. 앱을 실행했을 때 권한 부여

일반 권한과 위험 권한 (마시멜로 API23부터)

- 위험 권한은 실행 시 권한 부여

•앞에서 만든 앱을 설치하고 나서 SMS를 수신하면 브로드캐스트 수신자의 onReceive() 메서드가 정상적으로 호출됨
•그런데 마시멜로 버전의 단말부터는 onReceive() 메서드가 호출되지 않음
•<= 마시멜로 버전부터 권한을 부여하는 방식이 약간 바뀌었기 때문

•마시멜로 버전부터는 권한을 일반 권한과 위험권한으로 나누었으며, **위험 권한의 경우에는 앱을 설치할 때가 아니라 앱이 실행될 때 사용자에게 권한을 부여할 것인지 물어보도록 변경**됨

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

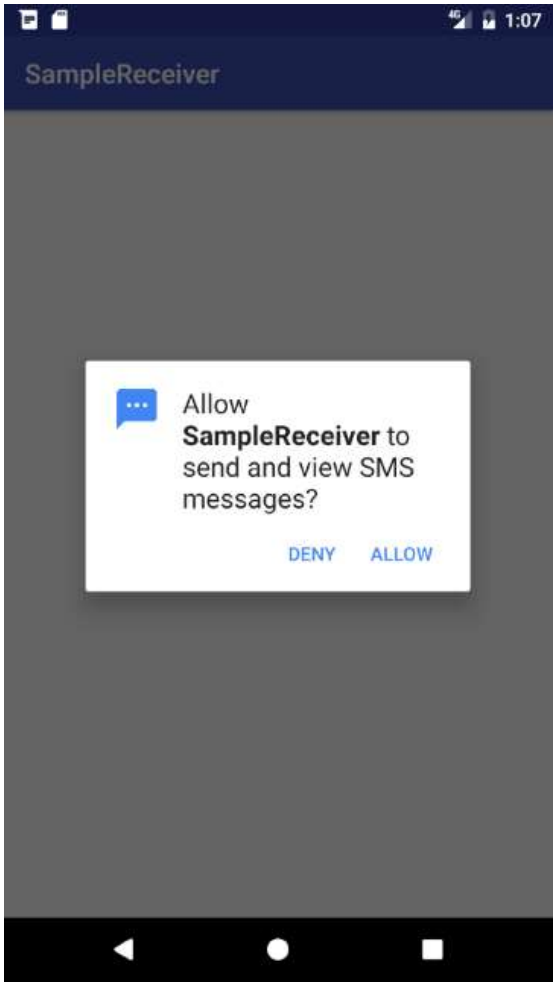
CALENDAR
- READ_CALENDAR
- WRITE_CALENDAR

SENSORS
- BODY_SENSORS

STORAGE
- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE

실행 시 권한 부여

- 실행 시 권한 부여를 묻는 대화상자 표시



위험권한을 사용할 수 밖에 없는 앱들은 앱이 실행될 때 권한을 부여해달라는 대화상자를 사용자에게 띄워야 함

위험권한인 RECEIVE_SMS를 부여해달라는 대화상자를 띄워주는 소스 코드 입력하기

매니페스트에 권한 추가

참조파일 SamplePermission>/app/manifests/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.techtown.permission">

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

    <application

중략...
```

위험 권한 부여 요청 코드 추가

```
public void checkPermissions(String[] permissions) {
  ArrayList<String> targetList = new ArrayList<String>();

  for (int i = 0; i < permissions.length; i++) {
    String curPermission = permissions[i];
    int permissionCheck = ContextCompat.checkSelfPermission(this, curPermission);
    if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
      Toast.makeText(this, curPermission + " 권한 있음.", Toast.LENGTH_LONG).show();
    } else {
      Toast.makeText(this, curPermission + " 권한 없음.", Toast.LENGTH_LONG).show();
      if (ActivityCompat.shouldShowRequestPermissionRationale(this, curPermission)) {
        Toast.makeText(this, curPermission + " 권한 설명 필요함.",
                       Toast.LENGTH_LONG).show();
      } else {
        targetList.add(curPermission);
      }
    }
  }

  String[] targets = new String[targetList.size()];
  targetList.toArray(targets);

  ActivityCompat.requestPermissions(this, targets, 101);
```

❷ 위험 권한 부여 요청하기

권한 요청 결과 확인 코드

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[],
                                       int[] grantResults) {
  switch (requestCode) {                                          ❶
    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();
      }

      return;
    }
  }
}
```

❷

위험 권한 자동 부여

참조파일 SamplePermission2>Gradle Scripts>build.gradle(Module:app)

```
중략…

allprojects {
  repositories {
    maven { url 'https://jitpack.io' }
  }
}

dependencies {

  중략…

  implementation 'com.github.pedroSG94:AutoPermissions:1.0.3'
}
```

위험 권한 자동 부여 코드 추가

참조파일 SamplePermission2>/app/java/org.techtown.permission2/MainActivity.java

```
public class MainActivity extends AppCompatActivity
                    implements AutoPermissionsListener {

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

    AutoPermissions.Companion.loadAllPermissions(this, 101); ---> ❶
  }

  @Override
  public void onRequestPermissionsResult(int requestCode, String permissions[],
                                         int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    AutoPermissions.Companion.parsePermissions(this, requestCode, permissions, this);
  }

  @Override
  public void onDenied(int requestCode, @NotNull String[] permissions) {
    Toast.makeText(this, "permissions denied : " + permissions.length,
                   Toast.LENGTH_LONG).show();
  }

  @Override
  public void onGranted(int requestCode, @NotNull String[] permissions) {
    Toast.makeText(this, "permissions granted : " + permissions.length,
                   Toast.LENGTH_LONG).show();
  }
}
```

❷

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

[매니페스트 파일의 기본 구조]

[Code]

```
<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>
```

스타일과 테마

- 대화상자의 경우 – 액티비티와 달리 타이틀 부분이나 모서리 부분의 형태가 약간 다르게 보이는데 이런 속성들을 다이얼로그 테마로 정의하여 액티비티에 적용하면 대화상자 모양으로 보이게 됨
- 안드로이드에서는 자주 사용되는 스타일과 테마를 제공하긴 하지만 필요에 따라 직접 정의해서 사용해야 함
- 스타일을 직접 정의하여 사용하고 싶다면 /res/values/style.xml 파일을 만들어야 함

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

-토스트의 모양을 XML 레이아웃으로 정의

메인 액티비티 코드 작성

-메인 액티비티에서 모양 설정

```
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

```
LayoutInflater 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="#ffffff00"
  />
 <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 XML 레이아웃 정의

-메인 액티비티의 레이아웃 정의

메인 액티비티 코드 작성

-메인 액티비티에서 대화상자 보여주기

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)

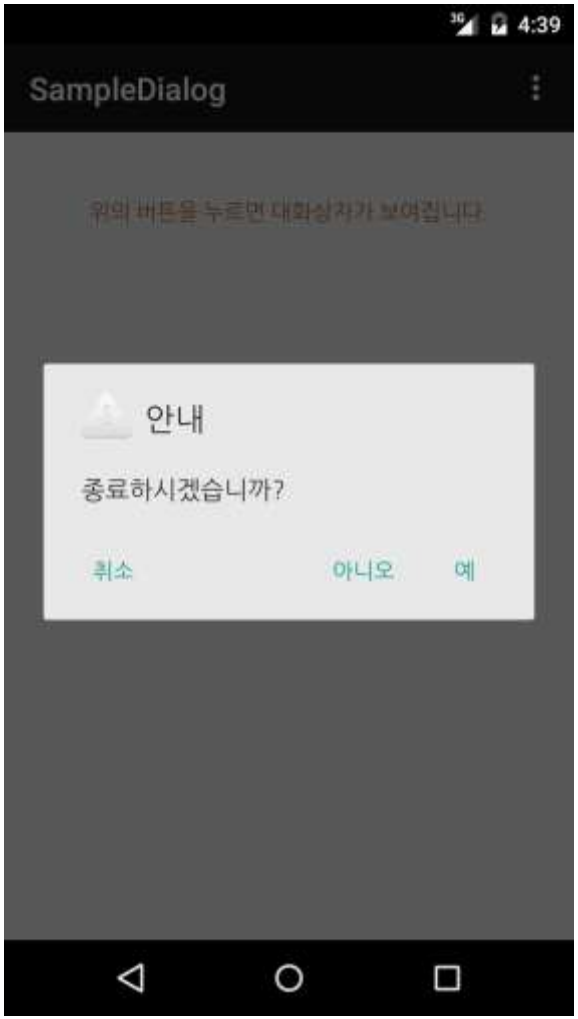
```
    builder.setPositiveButton("예", new DialogInterface.OnClickListener()
      public void onClick(DialogInterface dialog, int whichButton)
        msg = "예 버튼이 눌렀습니다. " + Integer.toString(whichButton);
            txtMsg.setText(msg);

    );

    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



```
public class DiaActivity extends AppCompatActivity {
    TextView txMsg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dia);
        txMsg = findViewById(R.id.textView);
    }

    public void onButtonClicked(View view) {
        AlertDialog dialog = createDialogBox();
        dialog.show();
    }

    private AlertDialog createDialogBox(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);

        builder.setTitle("안내");
        builder.setMessage("종료하시겠습니까?");
        builder.setIcon(R.drawable.alert_dialog_icon);
            builder.setPositiveButton("예", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                    String msg = "예 버튼이 눌렀습니다. " + Integer.toString(whichButton);
                    txMsg.setText(msg);
                }
            }
        );
```

```
import androidx.appcompat.app.AlertDialog;
```

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

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

    AlertDialog dialog = builder.create();

    return dialog;
  }


}
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="띄우기"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:onClick="onButtonClicked"/>
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp"
        android:text="버튼을 누르면 대화상자가 뜹니다."
        android:textSize="25sp"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

[References]

- 기본 서적

  2019, 정재곤, “Do it! 안드로이드 앱 프로그래밍(개정6판)”, 이지스퍼블리싱(주)

- Android Website

  http://www.android.com/

- Google Developer’s Conference

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

- Android SDK Documentation