



Android Programming

Components

Android Components

: At a glance

- ▶ 안드로이드 4대 컴포넌트
 - ▶ Activity
 - ▶ Service
 - ▶ Broadcast Receiver
 - ▶ Content Provider



주요 컴포넌트

: 4대 컴포넌트

▶ 액티비티 (Activity)

- ▶ 안드로이드 애플리케이션이 실행되는 기본 단위(보통, 하나의 화면을 의미)
- ▶ 사용자와 상호 작용하는 작업을 수행
- ▶ 사용자와 상호 작용하는 화면이 무엇인지 정의
- ▶ 상호작용을 처리할 기능을 구현
- ▶ 안드로이드 앱을 개발할 때, 각 화면마다 액티비티를 정의하고 구현하는 것이 일반적
- ▶ 하나의 안드로이드 앱은 하나 이상의 액티비티들로 구성
- ▶ 안드로이드 애플리케이션이 실행될 때, 처음 실행되는 메인 액티비티를 선택할 수 있다
 - ▶ 메인 액티비티 : 해당 안드로이드 앱이 실행되는 진입점
- ▶ 생애 주기(Life Cycle)에 따른 다양한 메서드들을 구현하게 된다

주요 컴포넌트

: 4대 컴포넌트

▶ 서비스(Service)

- ▶ 서비스는 사용자와 직접 상호작용을 하는 액티비티와는 달리, 화면에 표시되지 않고 백그라운드에서 작업을 수행
- ▶ 예) 음악 재생 앱, 스케줄링 앱

▶ 브로드캐스트 리시버(Broadcast Receiver)

- ▶ 안드로이드 내부의 특별한 이벤트를 받아야 하는 경우 구현
- ▶ 주로 시스템 상태와 관련된 시스템 공지
 - ▶ 배터리 부족, 언어 변경, 네트워크 활성화/비활성화 등
- ▶ 앱에서 특정 작업이 완료되었을 때 처리할 동작 구현에도 사용
- ▶ 애플리케이션에서 특정 이벤트를 받아 처리해야 할 필요가 있는 경우, 브로드캐스트 리시버를 구현하고 등록 설정을 하게 된다

주요 컴포넌트

: 4대 컴포넌트

- ▶ 콘텐츠 프로바이더 (Content Provider)
 - ▶ 어플리케이션 내의 데이터를 다른 어플리케이션과 공유할 수 있게 하는 컴포넌트
 - ▶ 안드로이드 앱 간에 데이터를 공유할 수 있는 방법을 제공한다
 - ▶ 예) 주소록 접근 등

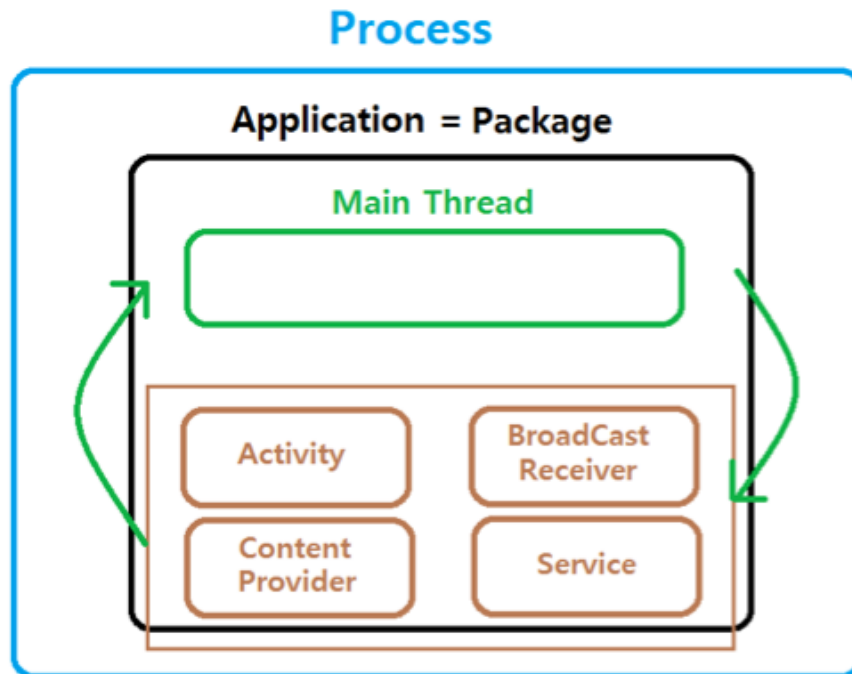
주요 컴포넌트

: 컴포넌트간 통신

- ▶ 인텐트 : 컴포넌트간 통신을 담당
- ▶ 액티비티, 서비스 및 브로드캐스트 리시버에 메시지를 전달할 때 사용
- ▶ 어떤 형식의 정보가 담겨 있는가에 따라 두 가지로 구분
 - ▶ 명시적 인텐트 (Explicit Intent)
 - ▶ 암시적 인텐트 (Implicit Intent)



Android App 실행 구조



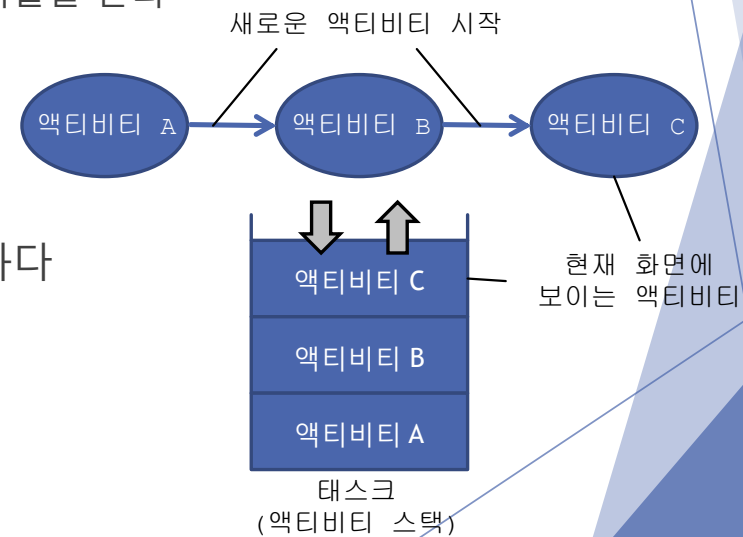
Activity

: 액티비티와 태스크

▶ 태스크

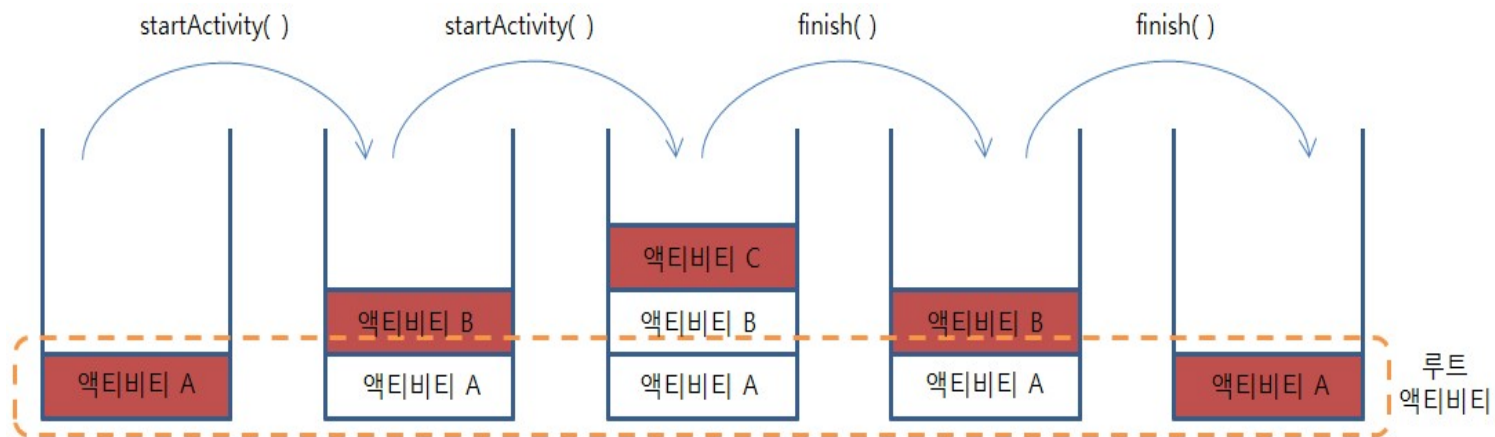
- ▶ 하나의 애플리케이션의 유기적인 연관 관계가 생성하는 액티비티들의 집합
- ▶ 태스크는 동일 프로세스 상으로 실행되는 액티비티들을 관리

- ▶ 하나의 액티비티가 다른 액티비티를 호출한다
- ▶ 액티비티 간에 경우에 따라 데이터 교환도 필요하다



Activity

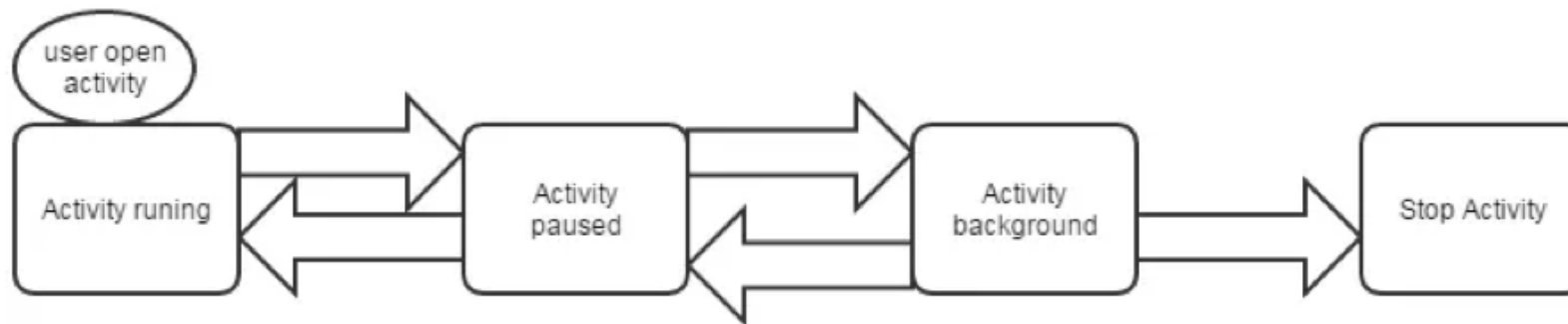
: 액티비티 스택



- ▶ startActivity() 메서드를 통해 새로운 액티비티를 시작할 수 있다.
- ▶ 파라미터로 Intent를 전달하여 데이터를 주고 받는다
- ▶ finish() 메서드는 명시적으로 자신(액티비티)를 종료할 때 사용할 수 있다

Activity

: 액티비티의 상태 (Activity State)



- ▶ **활성(Active, Running)** : 액티비티가 현재 화면에 표시되고 있는 상태
 - ▶ 사용자와 상호 작용이 가능, 대부분 이 상태에서 동작
- ▶ **일시정지(Paused)** : 액티비티가 화면에 표시되고 있지만, 상호작용을 못하는 상태
 - ▶ 이 경우, 시스템 메모리가 부족한 경우 강제 종료될 수 있다
- ▶ **정지(Stopped)** : 액티비티가 다른 액티비티에 의해 가려지거나, 화면상에 안보이는 경우
 - ▶ 메모리 부족한 경우, 일시정지 상태의 액티비티보다 강제로 종료될 가능성 높음

Activity

: 액티비티 생애주기 (Activity LifeCycle)

- ▶ 액티비티의 효율적 관리를 위해 액티비티의 상태 변화가 일어날 때마다 생애주기 메서드(콜백 메서드)를 호출한다

콜백 메서드	발생 시점
<code>onCreate()</code>	액티비티 생성시
<code>onStart()</code>	화면에 표시될 때
<code>onResume()</code>	활성화가 되기 직전
<code>onPause()</code>	화면에는 표시되나 상호작용을 할 수 없는 상태
<code>onStop()</code>	액티비티가 화면에 보이지 않을 때
<code>onRestart()</code>	정지상태로 대기 중이던 액티비티가 다시 호출될 때
<code>onDestroy()</code>	액티비티가 소멸될 때

Activity

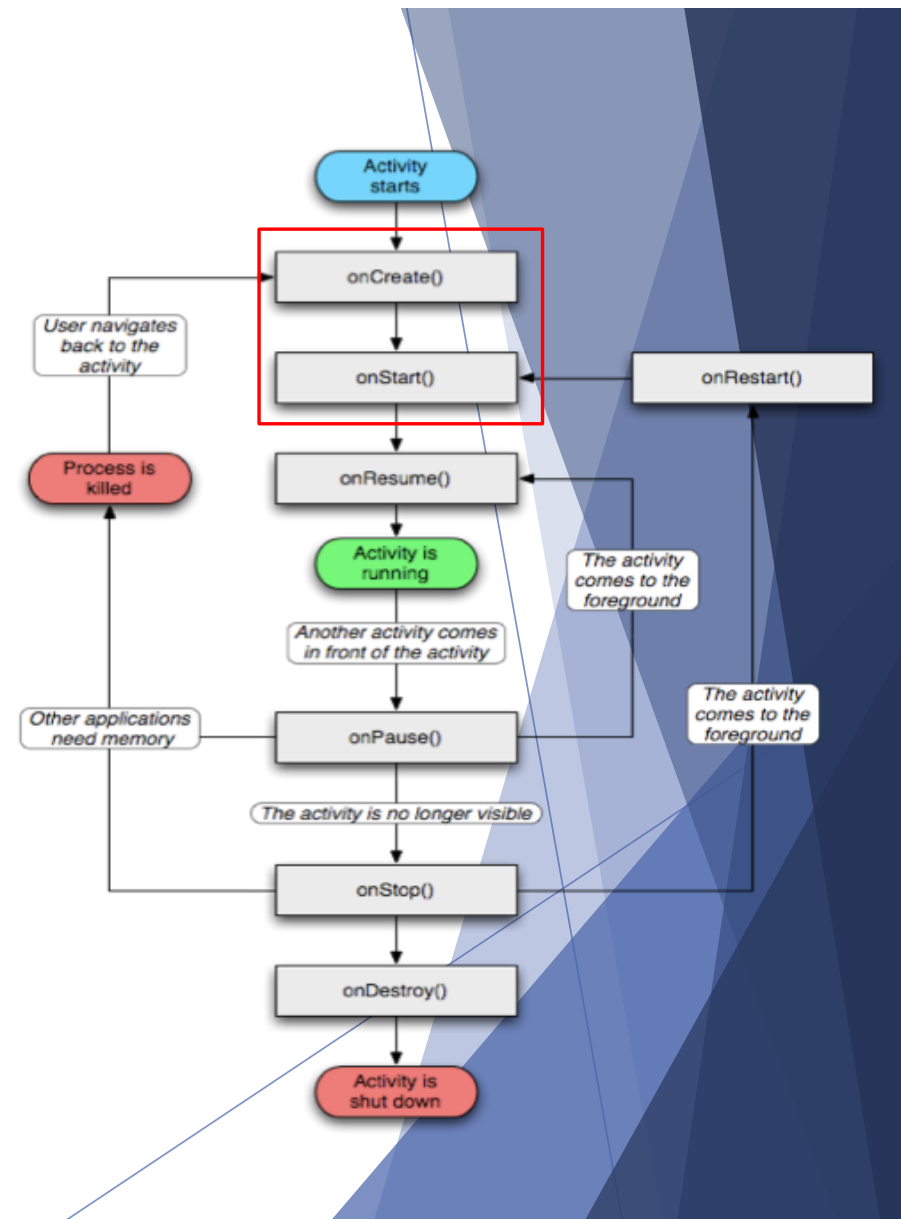
: 액티비티 생애주기 (Activity LifeCycle)

▶ onCreate()

- ▶ 액티비티가 생성될 때 처음으로 호출
- ▶ 전역 상태의 모든 리소스들을 초기화
예) layout, data binding 등
- ▶ 액티비티는 기본적으로 위젯들이 배치되어 있는 레이아웃을 구성하고, 위젯들이 사용자와 상호작용하는 코드를 포함

▶ onStart()

- ▶ 액티비티 초기화 과정을 마친 후, 사용자에게 보여줄 준비가 되었을 때 호출



Activity

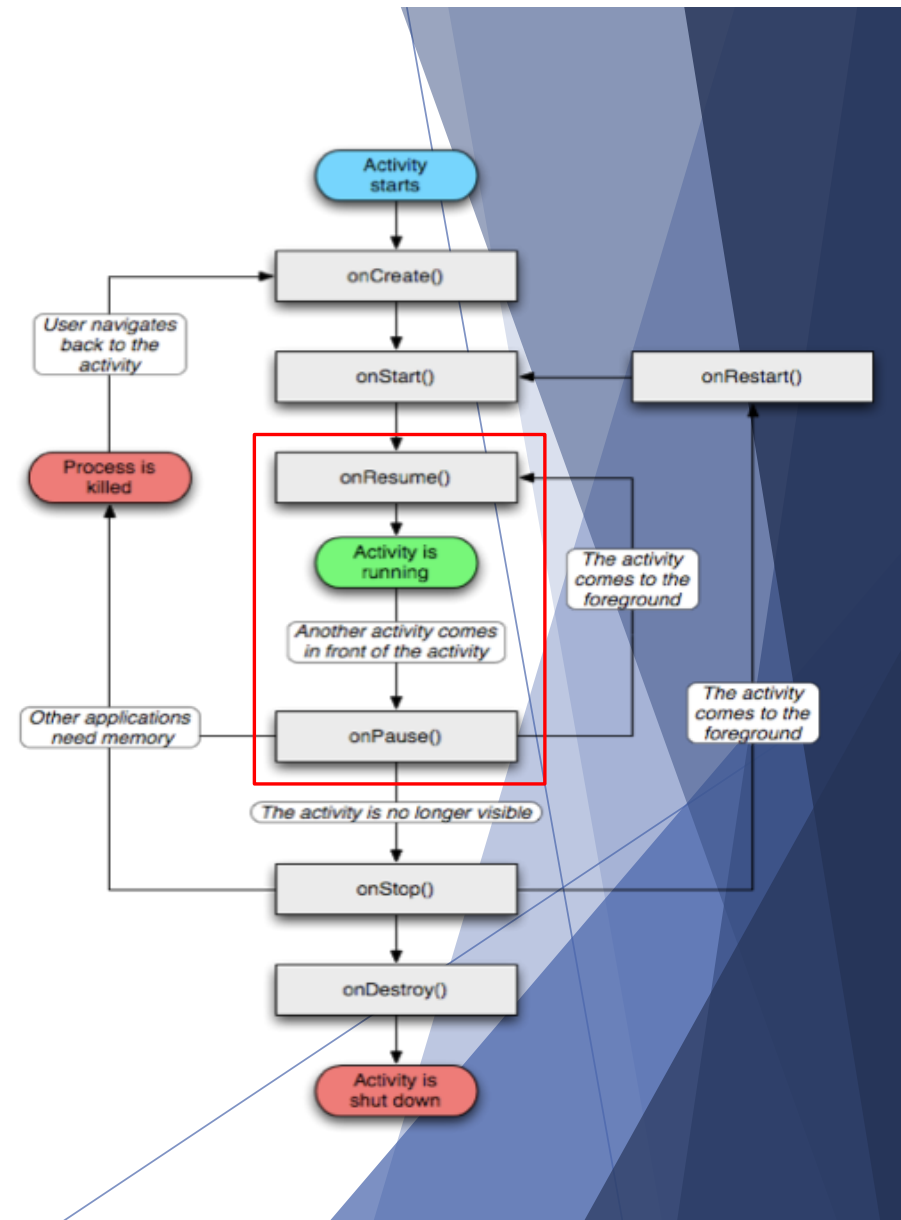
: 액티비티 생애주기 (Activity LifeCycle)

▶ onResume()

- ▶ 액티비티가 사용자에게 보여지고 사용자의 입력을 처리할 수 있음
- ▶ 액티비티 스택의 최상위에 위치

▶ onPause()

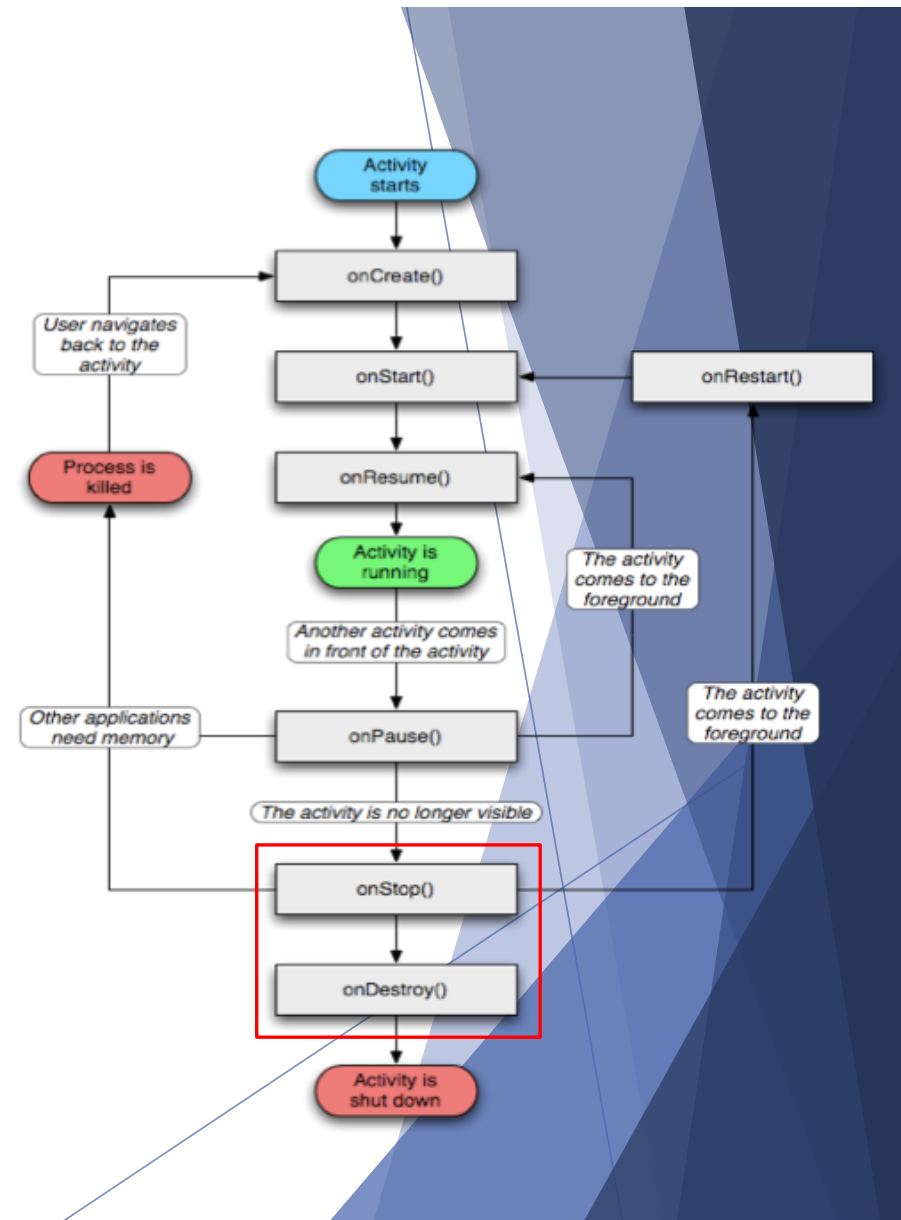
- ▶ 절전 상태 혹은 새로운 액티비티가 시작될 경우
- ▶ 포커스를 잃음
- ▶ 재개(resume) 되기 전 데이터 저장, 애니메이션 중지, 프로세서를 소비하는 작업 중단



Activity

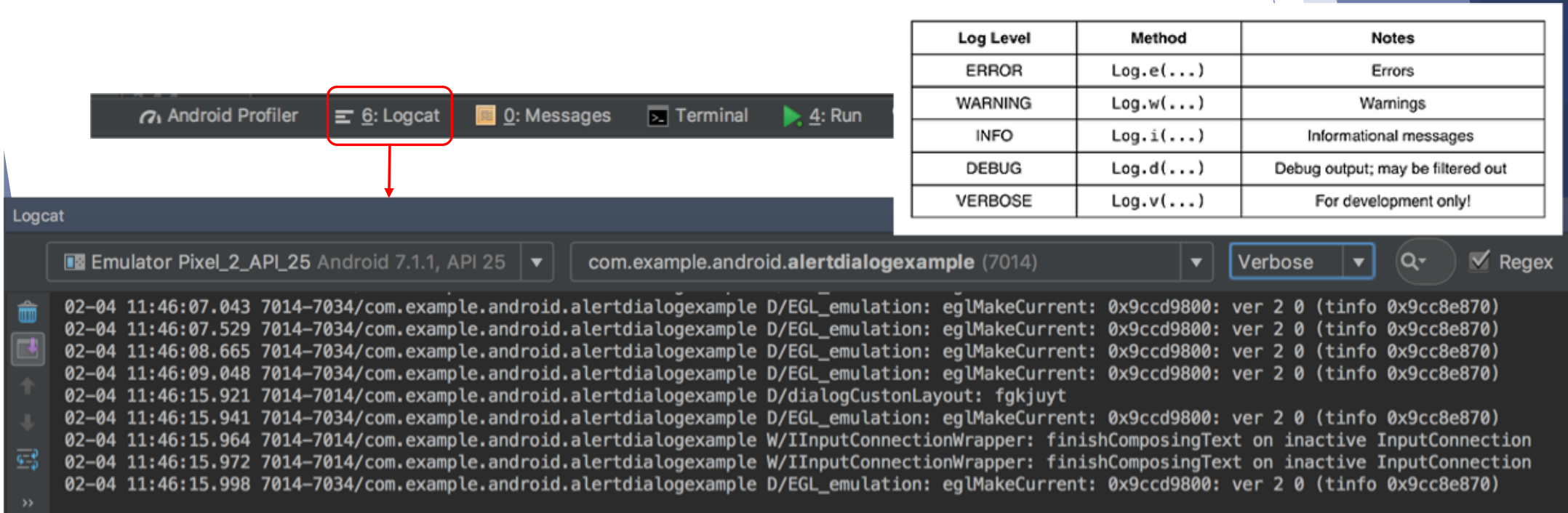
: 액티비티 생애주기 (Activity LifeCycle)

- ▶ **onStop()**
 - ▶ 더이상 액티비티가 사용자에게 보여지지 않음
 - ▶ 더 이상 액티비티 스택의 최상위에 위치하지 않음
- ▶ **onDestroy()**
 - ▶ 존재하는 모든 리소스를 해제
 - ▶ 시스템 내에 액티비티가 존재하지 않을 때
 - ▶ 명시적으로 `finish()` 가 호출될 때



Logcat를 이용한 로그 작성

- ▶ Logcat 모니터는 시스템 메시지 뿐 아니라 앱에 추가할 수 있는 메시지도 표시
 - ▶ 예) `Log.d(tag, message)` -> 디버그 메시지에 tag를 붙여 출력



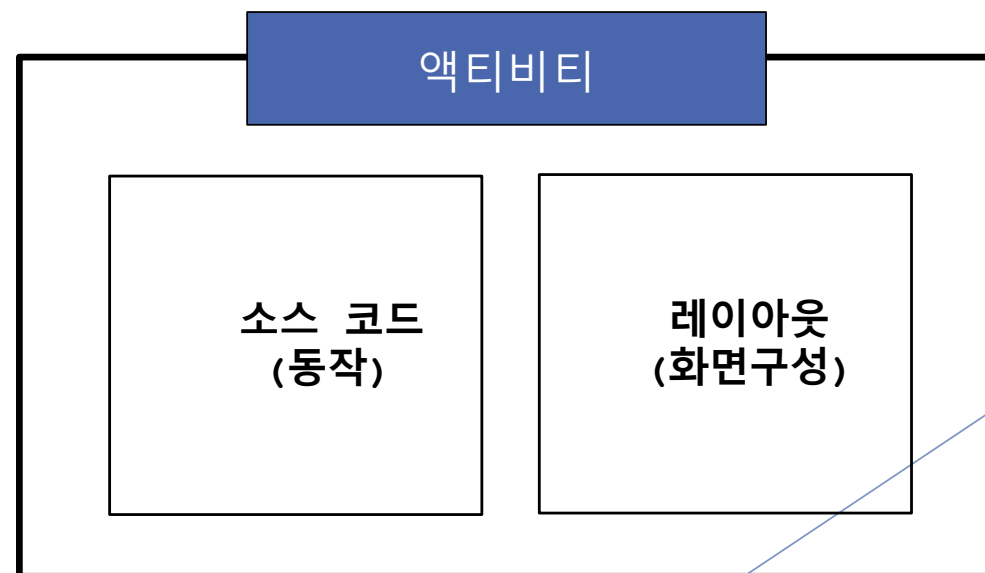
Log Level	Method	Notes
ERROR	<code>Log.e(...)</code>	Errors
WARNING	<code>Log.w(...)</code>	Warnings
INFO	<code>Log.i(...)</code>	Informational messages
DEBUG	<code>Log.d(...)</code>	Debug output; may be filtered out
VERBOSE	<code>Log.v(...)</code>	For development only!

```
02-04 11:46:07.043 7014-7034/com.example.android.alertdialogexample D/EGL_emulation: eglMakeCurrent: 0x9ccd9800: ver 2 0 (tinfo 0x9cc8e870)
02-04 11:46:07.529 7014-7034/com.example.android.alertdialogexample D/EGL_emulation: eglMakeCurrent: 0x9ccd9800: ver 2 0 (tinfo 0x9cc8e870)
02-04 11:46:08.665 7014-7034/com.example.android.alertdialogexample D/EGL_emulation: eglMakeCurrent: 0x9ccd9800: ver 2 0 (tinfo 0x9cc8e870)
02-04 11:46:09.048 7014-7034/com.example.android.alertdialogexample D/EGL_emulation: eglMakeCurrent: 0x9ccd9800: ver 2 0 (tinfo 0x9cc8e870)
02-04 11:46:15.921 7014-7014/com.example.android.alertdialogexample D/dialogCustomLayout: fgkjuyt
02-04 11:46:15.941 7014-7034/com.example.android.alertdialogexample D/EGL_emulation: eglMakeCurrent: 0x9ccd9800: ver 2 0 (tinfo 0x9cc8e870)
02-04 11:46:15.964 7014-7014/com.example.android.alertdialogexample W/IIInputConnectionWrapper: finishComposingText on inactive InputConnection
02-04 11:46:15.972 7014-7014/com.example.android.alertdialogexample W/IIInputConnectionWrapper: finishComposingText on inactive InputConnection
02-04 11:46:15.998 7014-7034/com.example.android.alertdialogexample D/EGL_emulation: eglMakeCurrent: 0x9ccd9800: ver 2 0 (tinfo 0x9cc8e870)
```

Activity

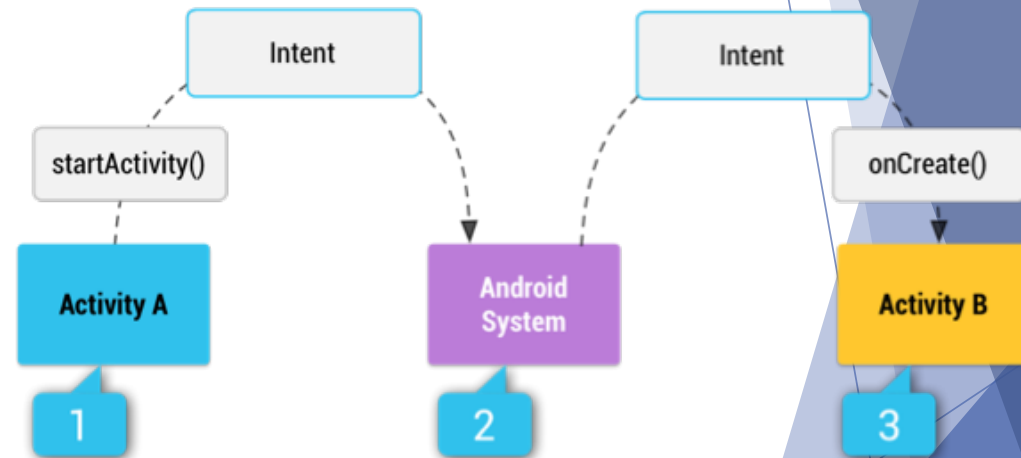
: [실습] 액티비티 추가하기

- ▶ 새 프로젝트를 생성하면 자동으로 액티비티가 하나 생성됩니다.
- ▶ 대부분의 앱들은 여러 액티비티로 구성되어 있습니다.
- ▶ 새로운 액티비티를 하나 더 추가하는 예제 `CreateActivity`를 작성해 봅시다
- ▶ 각 생명주기마다 `Log`를 출력, 생명주기의 실행 순서를 알아봅시다.



Intent

- ▶ 안드로이드의 메시지 전달 매커니즘
- ▶ 컴포넌트들 간에 서로 호출하기 위해서 전달해야 하는 비동기식 메시지
- ▶ 메시지 내에는 호출 대상 컴포넌트의 이름이 직접 명시되어 있을 수도 있고, 혹은 처리해 줘야 하는 작업과 데이터 등이 들어갈 수도 있다
- ▶ 호출된 액티비티가 자신을 호출한 액티비티에게 결과 값을 전달할 때도 Intent 객체에 데이터를 담아 전달
- ▶ 담겨있는 정보의 형태에 따라 명시적 인텐트 (Explicit Intent)와 암시적 인텐트(Implicit Intent)로 구분



Intent

: 인텐트가 포함할 수 있는 정보의 종류

- ▶ 컴포넌트의 이름
 - ▶ 호출하거나 메시지를 보낼 컴포넌트의 이름
 - ▶ 명시적 인텐트에서는 인텐트를 받을 대상 컴포넌트를 직접 지정

```
Intent intent = new Intent( this, NewActivity.class )
```

Intent

: 인텐트가 포함할 수 있는 정보의 종류

▶ 액션(Action)

- ▶ 인텐트를 통해 수행할 동작을 지정
- ▶ 브로드캐스트 메시지(브로드캐스트 리시버가 수신할 인텐트)의 경우 특정 상태를 의미
- ▶ 인텐트는 하나의 액션만을 가질 수 있으며 사용자 정의 액션을 만들어 사용할 수 있다

▶ 예)

- ▶ `android.intent.action.CALL` (액티비티, 전화를 건다)
- ▶ `android.intent.action.EDIT` (액티비티, 편집한다)
- ▶ `android.intent.action.MAIN` (액티비티, 테스트의 첫 액티비티로 액티비티를 시작)
- ▶ `android.intent.action.BATTERY_LOW` (브로드캐스트 리시버, 배터리 수준 낮음)

```
Intent intent = new Intent( android.intent.action.CALL, Uri.parse( "tel:010-4761-6934" ) )
```

Intent

: 인텐트가 포함할 수 있는 정보의 종류

- ▶ 카테고리
 - ▶ 액션이 실행하는데 필요한 추가 정보를 제공 (액션을 보충해주는 속성)
 - ▶ 인텐트는 여러 개의 카테고리를 가질 수 있다
- ▶ 예)
 - ▶ `android.intent.category.HOME` (홈화면에 표시되는 액티비티)
 - ▶ `android.intent.category.LAUNCHER` (Application Launcher에 표시되어 태스크의 루트 액티비티로 실행됨을 의미)

<https://developer.android.com/reference/android/content/Intent.html>

Intent

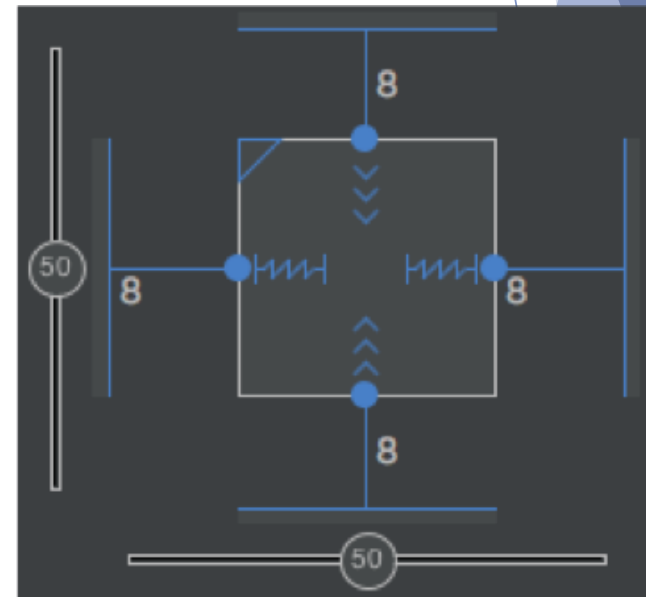
: 인텐트가 포함할 수 있는 정보의 종류

- ▶ 데이터 : **setData** 로 Intent 에 넣음
 - ▶ 처리할 데이터 전달 가능
 - ▶ 데이터는 URI 형태로 구성
 - ▶ 데이터 종류에 따라 타입이 다르다
- ▶ 예)
 - ▶ URL : <http://www.google.com> (없음, 인터넷 주소)
 - ▶ 미디어 : content://media/external/images/media/1 (image/jpeg, 이미지)
 - ▶ 미디어 : content://media/external/images/audio/1 (audio/mp3, 오디오)
 - ▶ 전화번호 : tel:010-1234-5678 (없음, 전화번호)
- ▶ 부가 정보 : **putExtra** 로 intent 에 넣음
 - ▶ URI 형식이 아닌 데이터를 전달할 때 사용

기본 레이아웃 다뤄보기

<https://academy.realm.io/kr/posts/cool-constraintlayout-droidcon-boston-2017/>

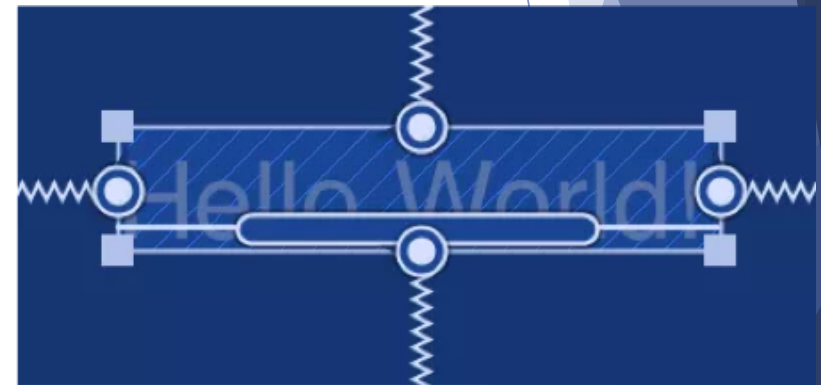
- ▶ 기존 버전에서는 LinearLayout과 RelativeLayout을 조합하여 화면을 구성
- ▶ 안드로이드 스튜디오 2.3 이후, 기본 레이아웃이 ConstraintLayout으로 변경
- ▶ ConstraintLayout
 - ▶ RelativeLayout과 유사, 상대적 위치에 따라 뷰의 배치를 제공
 - ▶ 새로워진 Layout Editor를 이용, 보다 강력하고 유연한 작업 가능



기본 레이아웃 다뤄보기

: 제약 조건

- ▶ ConstraintLayout은 뷰의 크기와 위치를 결정할 때 제약 조건(Constraint)을 사용
- ▶ 제약조건: 뷰가 레이아웃 안의 다른 요소와 어떻게 연결되는지 알려주는 조건
 - ▶ 뷰의 연결점(Anchor Point)과 대상(Target)을 연결
- ▶ Target
 - ▶ 같은 레이아웃 안에 들어있는 다른 뷰의 연결점
 - ▶ 부모 레이아웃의 연결점
 - ▶ 가이드라인
- ▶ 대상 뷰 - 타겟의 연결점
 - ▶ Top, Bottom, Left(Start), Right(End) - 위, 아래, 왼쪽, 오른쪽
 - ▶ CenterX, CenterY - 각 축의 중간
 - ▶ Baseline - 텍스트 기준 라인 (텍스트를 보여주는 뷰만 적용)



간단한 메시지의 출력

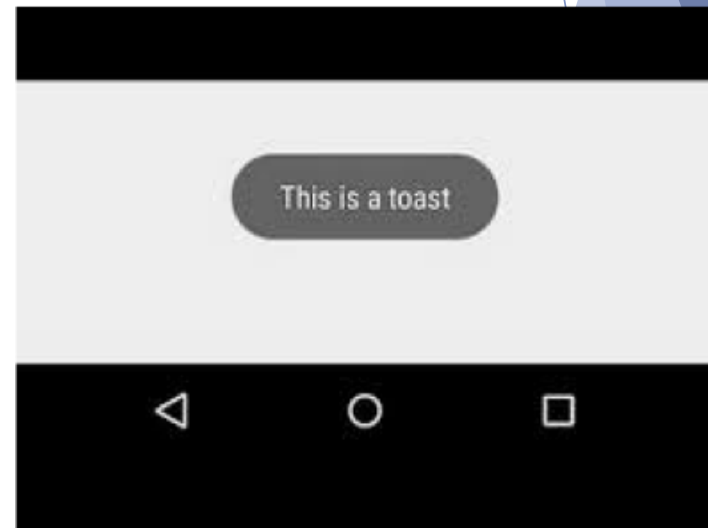
: Toast

- ▶ 화면에 잠시 나타났다가 사라지는 간단한 메시지의 출력

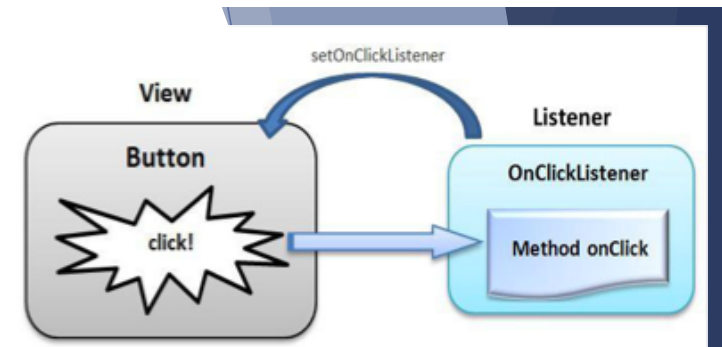
```
Toast.makeText({Context},  
    {출력 메시지},  
    {duration})  
    .show();
```

duration은 간단히 다음 두 값 중 하나로도 설정할 수 있다

- Toast.LENGTH_LONG
- Toast.LENGTH_SHORT



Event Listener의 등록



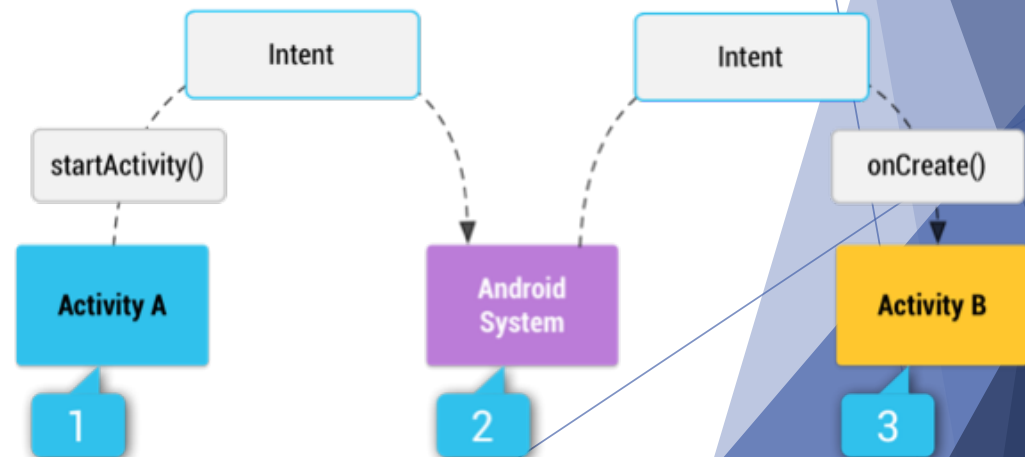
- ▶ 안드로이드의 위젯들은 이벤트를 발생시키며, 이 이벤트의 동작을 코드상에서 구현해 주어야 한다.

```
View btnNewActivity = findViewById(R.id.btnNewActivity);  
btnNewActivity.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        Toast.makeText(MainActivity.this,  
            "Button Clicked",  
            Toast.LENGTH_SHORT).show();  
  
        Intent intent = new Intent(MainActivity.this,  
            NewActivity.class);  
        startActivity(intent);  
    }  
});
```

다른 Activity의 실행

: startActivity

- ▶ 다른 Activity를 실행하기 위해 인텐트를 시스템에 의뢰하는 메서드
- ▶ 이 메서드에 실행하고자 하는 컴포넌트의 정보를 담은 Intent를 매개변수로 넘겨준다



Intent

: 명시적 인텐트

▶ 명시적 인텐트

- ▶ 호출하거나 메시지를 보낼 대상의 컴포넌트 이름이 지정되어 있는 인텐트
- ▶ 대상 컴포넌트에 인텐트 필터가 정의되어 있지 않아도 컴포넌트 호출과 메시지 전송을 할 수 있다

▶ [실습]

- ▶ 명시적 인텐트 사용하기 - `ExplicitIntent`
- ▶ 실습 `CreateActivity`를 조금 변경하여 `NewActivity`가 호출 가능하도록 변경합니다.

Intent

: 암시적 인텐트

▶ 암시적 인텐트

- ▶ 명시적 인텐트와 달리 **액션, 카테고리, 데이터**와 같은 특징을 포함하고 있는 인텐트
- ▶ 해석 과정이 필요 (Intent Resolving)
- ▶ 해석 과정에서 인텐트 조건에 맞는 컴포넌트를 찾기 위해 각 컴포넌트에 정의된 인텐트 필터를 검색
- ▶ 어떤 컴포넌트가 암시적 인텐트를 받으려면 반드시 인텐트 필터를 정의

Intent

: 인텐트 필터

- ▶ 인텐트 내의 여러 정보를 바탕으로 가장 적절한 대상 컴포넌트를 찾는 과정을 인텐트 해석(Intent Resolving)이라 함
- ▶ Intent Resolving 과정에서는 인텐트 내의 정보와 각 컴포넌트의 정보를 비교한다
- ▶ 비교를 위해 각 컴포넌트는 자신이 받을 수 있는 인텐트의 종류를 매니페스트에 정의해야 한다. 이를 인텐트 필터(Intent Filter)라 한다
- ▶ 안드로이드 내부에서는 수많은 애플리케이션에 수많은 인텐트가 발생하고 있으며 대부분 암시적 인텐트이다

Intent

: 암시적 인텐트를 사용해야 하는 이유

- ▶ 명시적 인텐트는 같은 앱 내의 컴포넌트 사이에서만 사용 가능
 - ▶ 다른 앱의 컴포넌트를 사용하려면 암시적 인텐트를 사용해야 한다
- ▶ 안드로이드 시스템에서 여러 앱(홈 화면, 다이얼러, 갤러리 등)의 호출은 모두 암시적 인텐트를 통해 이루어지므로 이를 대체하는 서드파티 앱 제작을 위해 암시적 인텐트를 사용해야 한다

Intent

: [실습] 암시적 인텐트의 사용

[실습] 암시적 인텐트 사용하기 (ImplicitIntent)

각 실행 버튼을 만들고 아래 암시적 인텐트를 테스트 해보세요.

1) 전화 걸기

```
Intent intent = new Intent( Intent.ACTION_CALL, Uri.parse("tel:000-0000") );
```

퍼미션 추가: <uses-permission android:name="android.permission.CALL_PHONE" />

2) 주소록 보기

```
Intent intent = new  
Intent( Intent.ACTION_PICK, Uri.parse("content://com.android.contacts/data/phones") );
```

3) 웹 이동

```
Intent intent = new Intent( Intent.ACTION_VIEW,  
Uri.parse("http://www.naver.com") );
```

Intent

: [실습]

[실습] 암시적 인텐트 사용하기 (cont'd)

4) 검색

```
Intent intent = new Intent(Intent.ACTION_SEARCH);  
intent.setPackage("com.android.browser");  
intent.putExtra("query", "android");
```

5) 위경도 정보값으로 지도 연결

```
Intent i = new Intent( Intent.ACTION_VIEW, Uri.parse("geo:38.899533,-  
77.036476" ) );
```


Permission

: Android 보안 아키텍처

- ▶ AndroidManifest.xml 에 들어가는 설정 - **uses-permission**
- ▶ 안드로이드는 기본적으로 애플리케이션에 다른 애플리케이션, 운영체제 혹은 사용자에게 안좋은 영향을 미칠 수 있는 작업을 수행할 수 있는 권한이 없다는 것을 바탕으로 디자인 (샌드박스: 실행 공간의 격리)
 - ▶ 사용자의 개인 데이터 읽기/쓰기, 다른 애플리케이션 파일 읽기/쓰기, 네트워크 액세스 수행 등
- ▶ 앱은 기본 샌드박스가 제공하지 않는 추가 기능에 대해 필요한 권한을 선언함으로써 앱간 리소스 및 데이터 공유를 할 수 있다

SMS 수신 모니터링을 위한 권한의 사용 예

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapplication">
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    ...
</manifest>
```

동적 권한 획득

- ▶ uses-permission은 일종의 신고제
 - ▶ 설치시 한번 물어보게 되며, 사용자가 거부할 수 없음
- ▶ 6.0 이후, 보안의 강화를 위해 사용자에게 위험 부담이 큰 권한은 거부할 수 있도록 재설계
- ▶ 동적 권한 획득 대상 Permission을 사용하고자 할 경우에는 사용자가 해당 권한을 체크 했는지 먼저 확인하는 코드를 삽입해야 한다

동적 권한 획득 대상 Permission 들 →

Table 1. Dangerous permissions and permission groups.

Permission Group	Permissions
CALENDAR	<ul style="list-style-type: none">• READ_CALENDAR• WRITE_CALENDAR
CAMERA	<ul style="list-style-type: none">• CAMERA
CONTACTS	<ul style="list-style-type: none">• READ_CONTACTS• WRITE_CONTACTS• GET_ACCOUNTS
LOCATION	<ul style="list-style-type: none">• ACCESS_FINE_LOCATION• ACCESS_COARSE_LOCATION
MICROPHONE	<ul style="list-style-type: none">• RECORD_AUDIO
PHONE	<ul style="list-style-type: none">• READ_PHONE_STATE• CALL_PHONE• READ_CALL_LOG• WRITE_CALL_LOG• ADD_VOICEMAIL• USE_SIP• PROCESS_OUTGOING_CALLS
SENSORS	<ul style="list-style-type: none">• BODY_SENSORS
SMS	<ul style="list-style-type: none">• SEND_SMS• RECEIVE_SMS• READ_SMS• RECEIVE_WAP_PUSH• RECEIVE_MMS
STORAGE	<ul style="list-style-type: none">• READ_EXTERNAL_STORAGE• WRITE_EXTERNAL_STORAGE

동적 권한 획득

: 동적 권한의 확인과
권한 요청

```
findViewById(R.id.btnCall).setOnClickListener((view) -> {  
    if (ActivityCompat.checkSelfPermission( context: MainActivity.this,  
        Manifest.permission.CALL_PHONE)  
        != PackageManager.PERMISSION_GRANTED) {  
        // 사용자가 권한을 허용한 상태  
        callPhone();  
    } else {  
        // 사용자가 권한을 허용하지 않은 상태  
        // 권한 획득 다이얼로그를 요청한다  
        ActivityCompat.requestPermissions( activity: MainActivity.this,  
            new String[] {Manifest.permission.CALL_PHONE},  
            requestCode: 200);  
    }  
}
```

▶ 권한의 체크

- ▶ `checkSelfPermission` 메서드를 이용, 권한을 체크한다
- ▶ 이 메서드의 반환값이 `PackageManager.PERMISSION_GRANTED`면 사용자가 권한을 승인한 상태 = 기능을 수행해도 된다

▶ 권한의 요청

- ▶ `requestPermission` 메서드에 요청할 권한을 설정한 후 호출한다. `requestCode`는 사용자 정의값

동적 권한 획득

: 권한 조정 결과의 확인

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                       @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {

    switch(requestCode) {
        case 200:
            if (grantResults.length > 0 &&
                grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // 권한 동의
                callPhone();
            } else {
                // 권한 동의 안함
                Toast.makeText( context: this,
                               text: "통화 권한이 없습니다",
                               Toast.LENGTH_SHORT)
                    .show();
            }
            break;
    }
}
```

- ▶ onRequestPermissionsResult 메서드를 오버라이드 하여 체크한다
 - ▶ 요청 결과의 값이 PackageManager.PERMISSION_GRANTED면 사용자가 권한 사용을 동의한 것

Intent

: [실습] 암시적 인텐트의 사용

[실습] 암시적 인텐트 사용하기 (`ImplicitIntent`) 수정

앞서 전화걸기 (`ACTION_CALL`)에서 발생했던 권한 오류를, `uses-permission`과 동적 권한 획득을 활용하여 정상적으로 작동하도록 수정해 보니다

Intent

: Intent Filter의 구성요소

- ▶ 필터의 액션은 인텐트의 액션과 일치해야 한다. 단, 인텐트의 액션이 생략되어 있으면 모두 통과하게 된다
- ▶ 인텐트 내의 카테고리를 검사하여 인텐트 필터에 정의된 카테고리와 일치 여부를 검사한다
- ▶ 카테고리는 반드시 정확하게 일치해야 통과 가능하다
- ▶ 인텐트에 카테고리를 생략할 경우 안드로이드 시스템에서는 `Intent.CATEGORY_DEFAULT` 를 추가한다
 - ▶ 따라서 액티비티가 암시적 인텐트를 받을 수 있으려면 인텐트 필터에 반드시 `CATEGORY_DEFAULT` 카테고리가 정의되어 있어야 한다

Intent

: Intent Filter의 예

```
<activity android:name="MainActivity">  
  <!-- This activity is the main entry, should appear in app launcher -->  
  <intent-filter>  
    <action android:name="android.intent.action.MAIN"/>  
    <category android:name="android.intent.category.LAUNCHER"/>  
  </intent-filter>  
</activity>
```

인텐트 필터는 여러 개 들어갈 수 있음

```
<intent-filter>  
  <action android:name="android.intent.action.EDIT"/>  
  <action android:name="android.intent.action.VIEW"/>  
  ...  
</intent-filter>
```

Intent

: [실습] 암시적 인텐트 적용해보기

ImplicitIntent2 앱에 ShareActivity를 외부 앱에서 암시적 호출이 가능하게 적용해 보는 실습 예제 입니다.

```
<activity  
  
    android:name="com.example.android.implicitintent2.  
    ShareActivity"  
        android:label="@string/app_name" >  
        <intent-filter>  
            <action  
            android:name="com.example.android.action.SHARE_ACTION" />  
            <category  
            android:name="android.intent.category.DEFAULT" />  
        </intent-filter>  
    </activity>
```

1. 인텐트 필터 추가

2. 호출

implicitIntent 앱에서 버튼을 추가하고
com.example.android.action.SHARE_ACTION 으로
호출하여 보세요

Intent

: [실습] 애플리케이션 구성 요소간 데이터 주고 받기

[실습] SendValExample

피호출 액티비티에게 데이터를 전달 하는 예제 입니다.

1. CallerActivity 구성
2. CalleeActivity 구성
3. 명시적 인텐트로 CallerActivity 가 CalleeActivity 호출
4. Caller 코드 추가

```
intent.putExtra("data_int", 10);  
intent.putExtra("data_String", "android");  
startActivity( intent );
```

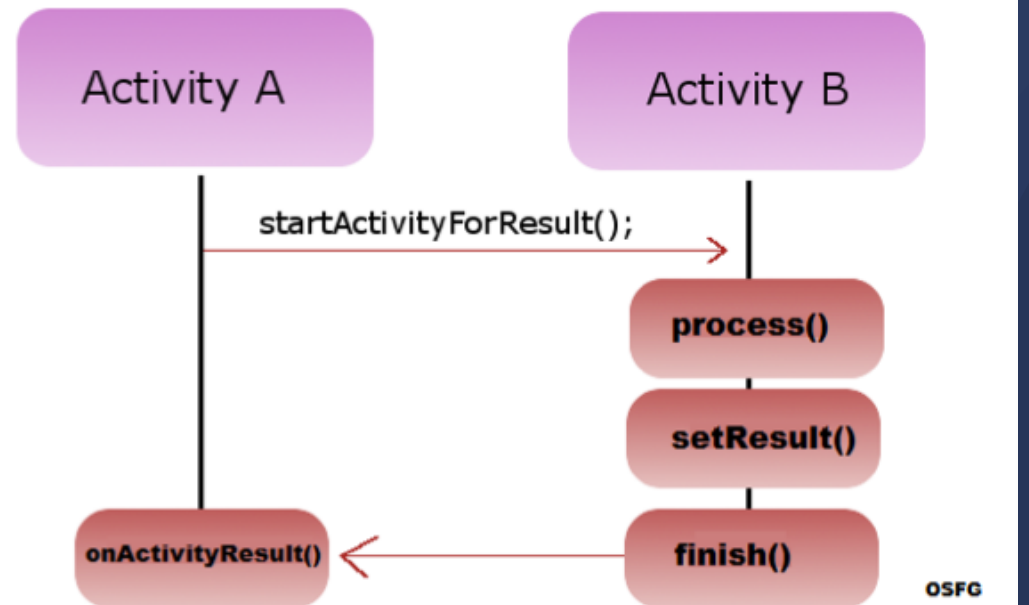
5. callee's onCreate(...)

```
Intent intent = getIntent();  
intent.getExtras().get("data_int")  
intent.getExtras().get("data_string")
```

결과값을 받기 위한 다른 Activity의 실행

: startActivityForResult

- ▶ 단지 Activity를 호출하는 것으로 끝나지 않고 호출한 액티비티로부터 결과값을 반환 받아야 할 경우 startActivityForResult를 이용하여 호출
- ▶ 결과를 반환하는 액티비티는 결과값을 설정한 후(setResult) 종료하면 호출한 액티비티에서 onActivityResult() 메서드로 데이터가 콜백(Callback) 된다.



결과값을 받기 위한 다른 Activity의 실행

: startActivityForResult [실습]

- ▶ 안드로이드의 Contacts 앱을 실행하여 선택한 사람의 전화번호를 가져와 Toast로 띄워봅시다
- ▶ SelectActivity를 만들고 버튼 세 개를 추가한 후(가위, 바위, 보) 선택한 버튼의 값을 MainActivity에서 받아서 Toast로 띄워 봅시다

