첫번째 예제

2014-08-23

이승진

**목차**

[1. 안드로이드 소개 2](#_Toc397605692)

[1) 안드로이드 시스템의 계층 구조 2](#_Toc397605693)

[2) API (Application Programming Interface) 2](#_Toc397605694)

[3) 안드로이드 어플리케이션의 주요 구성요소 3](#_Toc397605695)

[2. 첫번째 예제 (E01Button) 4](#_Toc397605696)

[1) 프로젝트 생성 4](#_Toc397605697)

[2) activity\_main.xml 수정 5](#_Toc397605698)

[3) MainActivity.java 수정 12](#_Toc397605699)

**학습목표**

기초 개념 소개

간단한 안드로이드 앱 만들기

이벤트 핸들러(event handler)와 리스너(listener) 개념 이해하기

# 안드로이드 소개

안드로이드(Android)는 Google이 만든 스마트폰용 운영체제이다.

Google은 안드로이드를 오픈소스로 공개하고있다.

안드로이드 운영체제는 리눅스(Linux)를 기반으로 만들어졌다.

그래서 운영체제의 핵심인 커널과 시스템의 주요 모듈은 C언어로 구현다.

그래픽 엔진도 역시 C언어로 구현되었다.

앱(App) 개발언어는 주로 Java이지만, 다른 언어로도 안드로이드 앱을 개발할 수 있다.

## 안드로이드 시스템의 계층 구조

|  |  |
| --- | --- |
| 계층 | 구현 언어 |
| 안드로이드 앱(App) | Java |
| 안드로이드 API 클래스 | Java |
| Java Virtual Machine (Dalvik) | C |
| 프로그래밍 라이브러리와 엔진 (Graphic, Animation, Database 등) | C |
| 운영체제 커널 (리눅스 기반) | C |
| 디바이스 드라이버 | C |
| 하드웨어 |  |

## API (Application Programming Interface)

system call, system programming라고 C는 배운다.

안드로이드 앱 개발을 공부한다는 것은 안드로이드 API를 공부한다는 말이다.

명령이나 리턴등을 정해놓은 집합.

API (Application Programming Interface)는 이름 그대로 애플리케이션을 프로그래밍하는데 필요한 인터페이스의 약자이다.

애플리케이션을 줄여서 앱이라고 한다.

인터페이스(Interface)는 두 시스템이 만나는 지점을 뜻한다.

앱은 운영체제 커널과 명령과 데이터를 주고 받아야 한다.

앱과 운영체제가 명령과 데이터를 주고 받기 위해서 만나는 지점 즉 인터페이스를   
운영체제의 Application Programming Interface라고 부른다.

앱을 C로 개발할 때는 API가 C 함수 형태로된 것을 사용하고,

앱을 Java로 개발할 때는 Java 클래스 형태로된 API를 사용한다.

앱은 안드로이드 API 클래스의 메소드를 호출해서 안드로이드 시스템에 명령과 데이터를 전달해야 한다.

앱을 개발하려면 안드로이드 API 클래스와 메소드를 공부해야 한다.

공부해야할 클래스와 메소드가 무척 많다.

http://developer.android.com/reference/packages.html

이 URL에 들어가보면, 안드로이드 API 클래스들을 묶어 놓은 패키지 목록을 볼 수 있다.

대략 200개 정도의 패키지가 나열되어 있다.

패키지 하나에 클래스가 10개만 들어있어도 200 x 10 = 1000

안드로이드 API 클래스 수가 대략 천개쯤이다.

안드로이드 프로그래밍을 공부한다는 것은 이 API 클래스들의 프로그래밍 방법을 이해하고 외운다는 말이다.

중요한 것으로만 골라서 배우겠지만 그래도 무척 많다.

많을뿐만 아니라 이해하기도 어렵다.

안드로이드 프로그래밍은 고등학교 수학 공부를 포함해서 지금까지 여러분이 공부했던 가장 어려운 과목보다도 여러 배 더 어려울 것이다.

## 안드로이드 어플리케이션의 주요 구성요소

### Activity

activity는 화면 하나에 해당하는 객체이다.

PC 운영체제에서 윈도우와 비슷한 개념이다.

그런데 PC에서는 화면에 동시에 여러 윈도우를 띄울 수 있지만,

화면이 좁은 안드로이드에서는 화면을 언제나 하나의 activity가 차지한다.

### Context

Activity 클래스의 부모 클래스이다.

### Service

화면에 보이지 않고 백그라운드에서 실행되는 서비스 객체이다.

### Content Provider

다른 어플리케이션에 데이터를 제공하기 위한 객체이다.

예를 들어 MP3 플레이어 앱은 다음 객체들로 구성된다

- 재생 화면을 구현한 activity 객체

- 노래 목록 화면을 구현한 activity 객체

- 웹브라우저 화면으로 넘어가도 백그라운드에서 계속 재생하기 위한 service 객체

주소록 앱은 다음 객체들로 구성된다

- 주소 목록 화면을 구현한 activity 객체

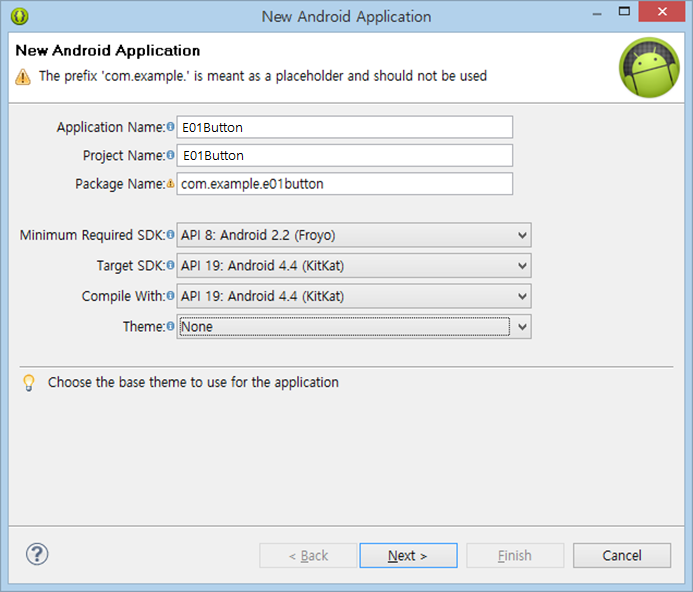
- 주소 편집 화면을 구현한 activity 객체

- 다른 앱에 주소 데이터를 제공하기 위한 content provider 객체

# 첫번째 예제 (E01Button)

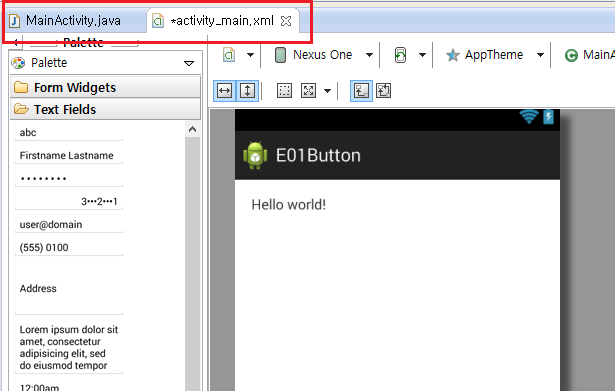
## 프로젝트 생성

이클립스 메뉴: File – New – Andoid Applicaton Project



위와 같이 입력하고 Next를 계속 눌러서 진행하자.

프로젝트가 생성되고 다음과 같이 편집창이 열리게 된다.



파일 이름을 읽어 보자. MainActivity.java와 activity\_main.xml 이다.

안드로이드에서 화면 하나에 해당하는 객체가 activity이다.

앱의 메인 화면을 구현할 activity가 자동으로 생성되었고 그 편집창이 열려있는 상태이다.

main activity 클래스의 소스코드가 MainActivity.java 파일에,

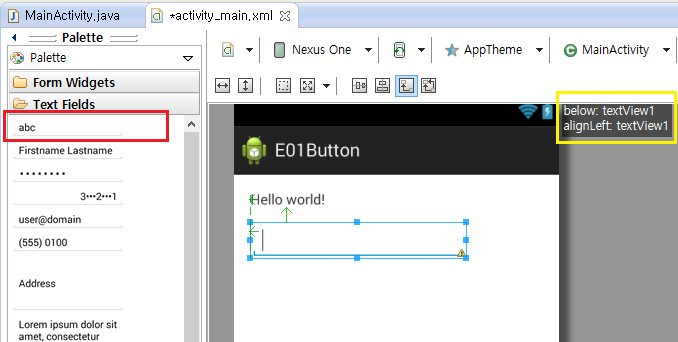
main activity 화면 모양을 정의하는 데이터가 activity\_,main.xml 파일에 들어있다.

액티비티는 이렇게 java 파일 하나와 xml 파일 하나로 구현된다.

## activity\_main.xml 수정

화면에 텍스트 박스와 버튼을 하나씩 추가해 보자.

### 텍스트박스 추가



화면 왼쪽 Palette에서 빨간색으로 표시한 부분이 문자열을 입력하기 위한 텍스트박스이다.

이 텍스트박스를 드래그하여 액티비티의 hello world! 아래에 드롭하자.

위 그림에서 텍스트박스 옆에 표시된 작은 녹색 화살표 두개를 주목해서 보자.

이 녹색 화살표는 텍스트박스의 위치가 hello world 문자열을 기준으로 그 아래이고,

또 역시 hello world 문자열을 기준으로 왼쪽 경계가 맞춰졌다는 표시이다.

녹색 화살표가 위 그림과 같이 보이도록 텍스트박스를 드래그하여 위치를 조정하자.

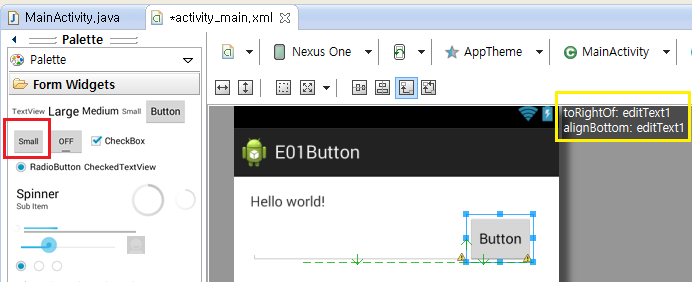
위 그림에서 화면 왼쪽에 노락색으로 표시한 부분을 주목해서 보자.

이것은 지금 현재 선택된 상태인 텍스트박스가 textView1 객체의 아래에(below)이고,

또 textView1 객체와 왼쪽 경계가 맞춰졌다(alignLeft)는 뜻이다.

위 그림에서 hello world! 로 보이는 객체의 id가 textView1 이다.

### 버튼 추가



화면 왼쪽 Palette에서 빨간색으로 표시한 부분이 작은 버튼이다..

이 버튼을 드래그하여 액티비티의 텍스트박스의 오른쪽에 드롭하자.

위 그림에서 버튼 옆에 표시된 작은 녹색 화살표들을 주목해서 보자.

이 녹색 화살표는 텍스트박스의 오른쪽에 버튼이 위치하고,

텍스트박스와 버튼의 아래쪽 경계가 맞춰졌다는 표시이다.

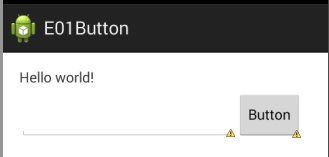
위 그림에서 화면 왼쪽에 노락색으로 표시한 부분을 주목해서 보자.ㄴ

이것은 지금 현재 선택된 상태인 버튼이 editText1 객체의 오른쪽에 위치하고,

또 editText1 객체와 아래쪽 경계가 맞춰졌다(alignBottom)는 뜻이다.

앞에서 추가한 텍스트박스의 id가 editText1 이다.

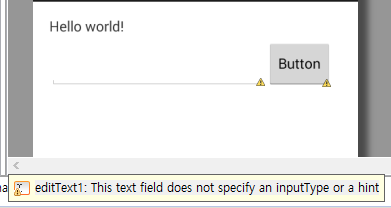
### 경고 메시지 보기



위 그림에서 노란색 세모 표시를 주목하자.

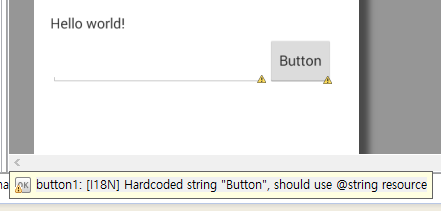
이 노란색 세모는 무엇인가 문제가 있다는 표시이다.

마우스 커서를 노란색 세모 위에 가만히 올려 놓으면 문제가 무엇인지 보인다.



editText1: This text field does not specify an inputType or a hint

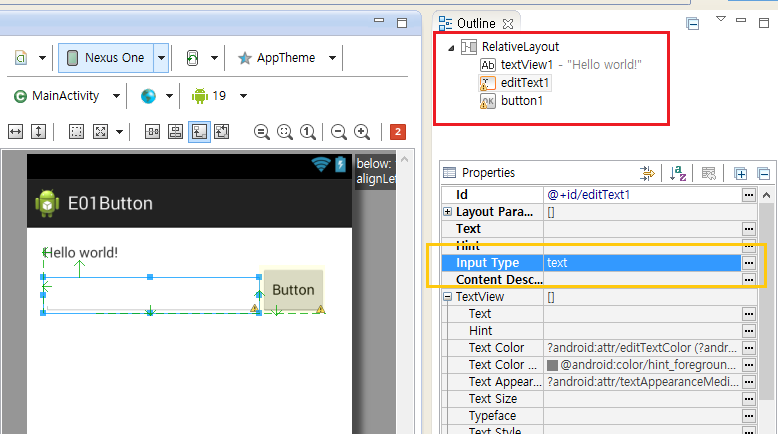
id가 editText1인 객체의 inputType 속성을 지정하지 않았다는 경고(warning)이다.



button1: [I18N] Hardcoded string "Button", should use @string resource

id가 button1인 객체의 어떤 속성을 "Button"이라고 직접 코딩(hard coding)했는데 @string 리소스를 사용하는 것이 좋다는 경고(warning)이다.

### 속성값 설정



위 그림에서 화면 오른쪽의 빨간색으로 표시한 Outline 창을 주목하자.

액티비티의 내용을 구성하는 객체들의 루트(root)는 RelativeLayout객체이고,

그 아래 textView1, editText1, button1 객체들이 들어있다는 것을 보여준다.

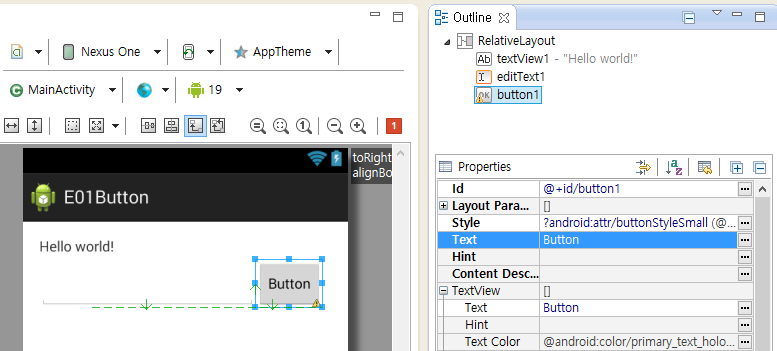
여기서 editText1을 클릭하면 위 화면과 같이 editText1이 선택된 상태로 표시된다.

그리고 화면 오른쪽의 노란색으로 표시한 Properties 창이 이 editText1의 속성(properties)을 보여준다.

inputType 속성을 지정하지 않았다는 노란색 세모 경고를 해결해보자.

editText1의 Properties 창에서 Input Type 속성을 클릭하여 text로 바꾸자.

InputType 속성값을 text로 값을 바꾸자마자 노란색 세모 경고 표시가 사라질 것이다.



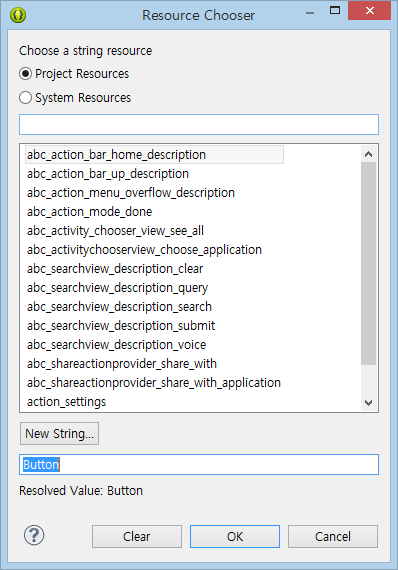
화면 왼쪽 Outline창에서 button1을 클릭하면 버튼이 선택된다.

Properties 창에서 버튼의 Text 속성값이 Button 문자열인 것일 볼 수 있다.

Text 속성값이 직접 문자열을 입력하는 것은 권장할만한 방법이 아니다.

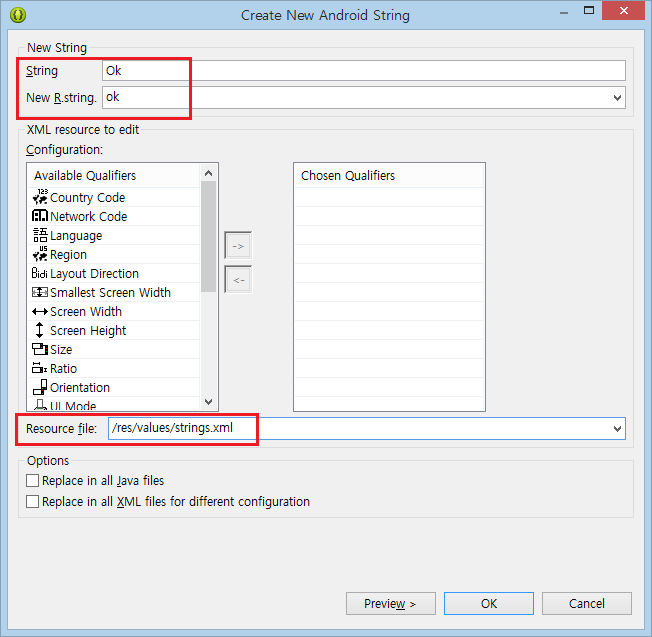
### 문자열 리소스(string resource) 만들기

Text 속성 오른쪽  작은 버튼을 클릭하자



문자열 리소스 목록을 보여주는 창이 열렸다.

이 창에서 New String 버튼을 클릭하자.



문자열 리소스(string resource)를 추가하기위한 창이 열렸다.

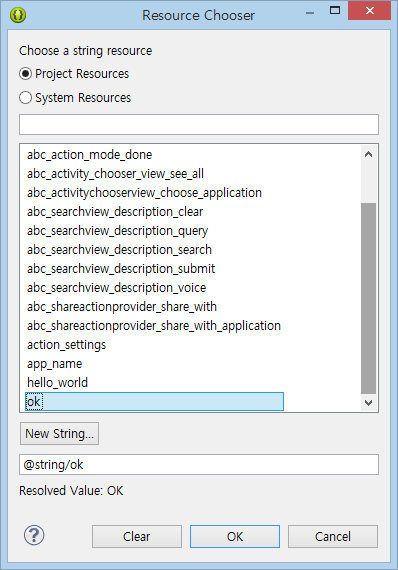
위 창에서 String 항목에 입력하는 내용이 바로 문자열 내용이다. 한글로 '확인'이라고 입력해도 좋다.

New R.tring 항목은 이 문자열 리소스의 id를 지정한다.

Resource file은 이 문자열 리소스가 입력될 xml 파일을 지정한다.

위와 같이 입력하고 Ok 버튼을 누르면 xml 파일에 문자열 리소스가 등록됨과 동시에 창이 닫힌다.

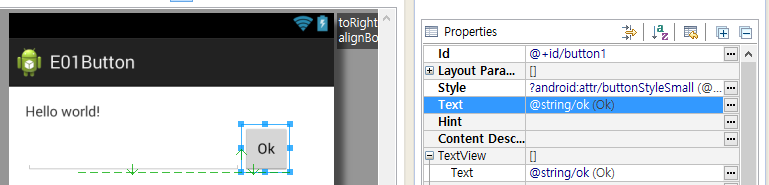
### 속성값으로 문자열 리소스 선택하기



방금 등록한 ok 문자열 리소스가 여기에 보인다.

여기에 보이는 것은 문자열 리소스의 내용이 아니고 id이다.

이것을 클릭하여 선택하고 Ok 버튼을 눌러서 창을 닫자.



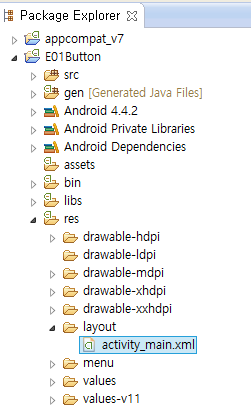
버튼에 표시된 문자열은 방금 우리가 등록한 문자열 리소스의 내용이다.

그리고 오른쪽 속성창에 표시된 @string/ok 는 바로 이 문자열 리소스의 id이다.

단축키 Ctrl+S를 눌러서 편집한 내용을 저장하고,

단축키 Ctrl+F4를 눌러서 파일을 닫자.

방금 우리는 액티비티의 화면 내용을 수정하고 저장하였다.



이 파일이 저장된 위치는 res 폴더 아래, layout 폴더 아래이고, 파일 이름은 activity\_main.xml 이다.

다시 편집하려면 이 파일을 더블클릭하여 열면 된다.

res 폴더는 리소스(resource) 파일들을 모아 놓은 폴더이다.

프로젝트의 구성요소인 데이터 파일들을 리소스 파일이라고 한다.

아이콘 파일, 비트맵 파일, 배경음악 파일, 화면 내용 파일들이 리소스 파일이다.

layout 폴더는 레이아웃 리소스(layout resource) 파일들을 모아 놓은 폴더이다.

화면의 내용 배치를 안드로이드에서는 레이아웃(layout)이라고 한다.

화면의 내용을 저장한 xml 파일을 레이아웃 리소스 파일이라고 한다.

## MainActivity.java 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | package com.example.e01button;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.view.View.OnClickListener;  import android.widget.Button;  import android.widget.EditText;  import android.widget.TextView;  public class MainActivity extends ActionBarActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);    Button b = (Button)findViewById(R.id.button1);  OnClickListener listenerObj = new OnClickListener() {  @Override  public void onClick(View v) {  EditText e = (EditText)findViewById(R.id.editText1);//아이디로 객체찾아 참조리턴  CharSequence s = e.getText();  TextView t = (TextView)findViewById(R.id.textView1);  t.setText(s);  }  };  b.setOnClickListener( listenerObj );  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

위와 같이 수정하자.

위 코드의 대부분은 자동으로 생성된 코드이고, 새로 추가된 부분은 (줄7~11), (줄20~30)이다.

(줄13) 화면을 담당하는 액티비티 클래스이다.

(줄15) 부모 클래스의 메소드를 재정의(override)하는 경우, 그 메소드의 앞에 붙여주는 어노테이션(annotation)이다.

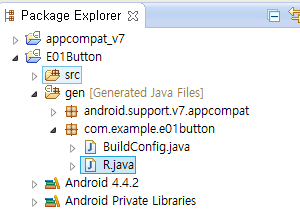
(줄16) 액티비티 초기 설정을 해야할 때, onCreate 메소드가 저절로 호출된다. 따라서 onCreate 메소드에 액티비티의 초기 설정을 구현한다.

(줄17) 부모 클래스의 onCreate 메소드를 호출한다.

자식 클래스에서 재정의한 부모 클래스의 메소드를, 자식 클래스의 내부에서 호출할 때에는 super 키워드를 사용해서 호출해야 한다.

(줄18) R.layout.activity\_main은 int 형 상수이다.

이 상수는 프로젝트를 생성할 때 자동으로 생성된 R.java 파일에 들어있다.



앞에서 화면 내용이 들어있는 res/layout/activity\_main.xml 파일을 수정하였다.

이 리소스 파일을 Java 소스코드에서 사용하려면 id가 필요하다.

이 리소스 파일에 대한 id는 프로젝트가 생성될 때 R.java 파일에 자동으로 생성되었다.

이 리소스 파일에 대한 id가 R.layout.activity\_main 이다.

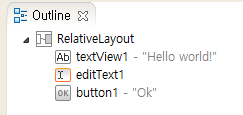
리소스 파일의 경로명과 유사한 형태로 id가 생성된다.

/res/layout/activity\_main.xml → 경로명

R.layout.activity\_main → id

(줄18) R.layout.activity\_main 레이아웃 리소스의 내용대로 액티비티 화면을 만들라는 메소드 호출이다.

이 파일의 내용대로 TextView, EditText, Button 객체가 자동으로 만들어진다.



앞에서 activity\_main.xml 파일을 편집할 때, 위와 같은 내용이 보였었다.

"Hello world!" 문자열이 채워진 TextView 객체와,

비어있는 EditText 객체와

"Ok" 문자열이 채워진 Button 객체가 (줄18)의 setContentView(R.layout.activity\_main) 메소드 호출에 의해서 생성된다.

생성된 이 객체들을 Java 코드에서 사용하려면, id가 필요하다.

이 객체들의 id는 각각 R.id.textView1, R.id.editText1, R.id.button1 이다.

이 id 상수들도 R.java에 이미 등록되어 있다.

(줄20) id가 R.id.button1인 Button객체에 대한 참조를 얻는 메소드 호출이다.

이 Button 객체는 줄18의 메소드 호출에 의해서 생성되었다.

안드로이드에서는 버튼이나 체크박스 같은 화면의 구성요소들을 뷰(View) 객체라고 부른다.

안드로이드에서만 이렇게 부른다. 안드로이드가 아닌 다른 시스템에서 뷰라고 하면 보통 화면 전체를 가르키는 뜻으로 사용하는데, 안드로이드만 독특하게 버튼 하나, 체크박스 하나 같은 작은 것들을 뷰라고 부른다.

findViewById 라는 메소드는, 파라미터로 주어진 id로 View 객체를 찾아서, 찾은 객체에 대한 참조를 리턴한다. 보통 객체를 리턴한다고 짧게 말하지만, 정확히 말하면 Java 언어에서 객체 자체는 절대 리턴될 수 없고, 객체에 대한 일종의 주소 같은 것인 참조만 리턴된다.

(줄20) findViewById가 리턴한 참조를 지역변수 b에 저장한다.

보통 변수에 객체를 저장한다고 짧게 말하지만, 정확히 말하면 Java 언어에서 객체 자체는 절대 어디에 저장될 수 없고, 객체에 대한 일종의 주소 같은 것인 참조만 저장된다.

### anonymous inner class 문법

(줄21~29) 리스너 객체를 하나 만드는 것을 annonymous inner class 문법으로 구현한 문장이다.

한방에 다 처리하기 위함.

이 Java 코드가 하는 일은 다음과 같다.

new OnClickListener() {

@Override

public void onClick(View v) {

EditText e = (EditText)findViewById(R.id.editText1);

CharSequence s = e.getText();

TextView t = (TextView)findViewById(R.id.textView1);

t.setText(s);

}

};

1) OnClickerListener 인터페이스(interface)를 상속(implements)하여 자식 클래스를 생성한다.

2) OnClickerListener의 onClick 메소드를 그 자식 클래스에서 **재정의(override)한다**

3) 그 자식 리스너 클래스의 객체를 하나 생성(new)한다.

(줄21~30)의 Java 코드가 하는 일은 다음과 같다.

OnClickListener listenerObj = new OnClickListener() {

@Override

public void onClick(View v) {

EditText e = (EditText)findViewById(R.id.editText1);

CharSequence s = e.getText();

TextView t = (TextView)findViewById(R.id.textView1);

t.setText(s);

}

};

b.setOnClickListener( listenerObj );

1) OnClickerListener 인터페이스(interface)를 상속(implements)하여 자식 클래스를 생성한다.

2) OnClickerListener의 onClick 메소드를 그 자식 클래스에서 재정의(override)한다

3) 그 자식 리스너 클래스의 객체를 하나 생성(new)한다.

4) 생성된 리스너 객체에 대한 참조를 지역변수 listenerObj에 저장한다.

5) 변수 b가 참조하는 Button 객체의 setOnClickListener 메소드를 호출한다

6) 이 메소드의 파라미터로 리스너 객체에 대한 참조를 전달한다

파라미터로 객체를 전달한다고 짧게 말하지만, 정확히 말하면 Java 언어에서 객체 자체는 절대 전달될 수 없고, 객체에 대한 일종의 주소 같은 것인 참조만 전달된다.

Java에서 객체는 생성된 후 결코 어디에 저장되거나 전달되거나 리턴되거나 할 수 없다. 그냥 생성된 위치에서 그대로 가만히 있다. 어디에 저장되고 전달되고 리턴되는 것은 객체 그 자체가 아니라 그 객체에 대한 일종의 주소 같은 것인 참조뿐이다.

### 이벤트 핸들러(event handler)

어떤 특정한 상황이나 사건이 벌어지자마자 자동으로 호출되어야 하는 메소드를 이벤트 핸들러(event handler) 메소드라고 부른다. 이벤트 핸들러 메소드의 이름은 on으로 시작하는 것이 관례이다.

예를들어 (줄16)의 onCreate 메소드, (줄23)의 onClick 메소드, (줄34)의 onCreateOptionsMenu 메소드, (줄40)의 onOptionsItemSelected 메소드가 이벤트 핸들러이다.

액티비티 클래스의 onCreate 이벤트 핸들러 메소드는 액티비티의 초기 설정을 해야될 때 자동으로 호출된다.

액티비티 초기 설정을 구현해야 한다면, 액티비티 클래스의 자식 클래스에서 onCreate 메소드를 재정의(override)해서 여기에 구현해야 한다.

액티비티 클래스의 onCreateOptionsMenu 이벤트 핸들러 메소드는 액티비티의 메뉴를 만들어야 할 때 자동으로 호출된다. 액티미티의 메뉴를 만드는 코드는, 액티비티 클래스의 자식 클래스에서 onCreateOptionsMenu 메소드를 재정의(override)해서 여기에 구현해야 한다.

액티비티 클래스의 onOptionItemSelected 이벤트 핸들러 메소드는 액티비티의 메뉴가 클릭되면 자동으로 호출된다. 액티미티의 메뉴가 클릭되었을 때 실행되야 할 코드는, 액티비티 클래스의 자식 클래스에서 onOptionItemSelected 메소드를 재정의(override)해서 여기에 구현해야 한다.

액티비티의 Create, CreateOptionsMenu, OptionItemSelected 이벤트가 발생했을 때 자동으로 호출되어 실행되어야 할 코드는, 액티비티의 자식 클래스를 만들어서 onCreate, onCreateOptionsMenu, onOptionItemSelected 이벤트 핸들러 메소드를 재정의하여 여기에 구현해야 한다.

이벤트가 발생하면 이벤트 핸들러 메소드가 즉시 자동으로 호출되므로, 이벤트 핸들러 메소드를 호출하는 코드는 작성할 필요없다.

### 리스너(Listener)

액티비티의 이벤트 핸들러는 액티비티의 자식 클래스를 만들고, 자식 클래스에서 이벤트 핸들러 메소드를 재정의하는 형태로 구현하였다.

그런데 어떤 클래스의 이벤트 핸들러는 자식 클래스를 만들지 않고, 대신 리스너(Listener) 클래스를 만들어서, 리스너 클래스에 이벤트 핸들러 메소드를 재정의하는 형태로 구현한다.

이렇게 오직 이벤트 핸들러를 구현하기 위한 목적의 클래스를 리스너 클래스라고 부른다. (event listener class)

Java에서 이벤트 핸들러 메소드 이름은 on으로 시작하는 것이 관례이고, 리스너 클래스 이름은 On으로 시작하는 것이 관례이다.

리스너 클래스들에 다형성을 적용하기 위한 목적으로 리스너 클래스들의 인터페이스(interface)가 안드로이드 API에 미리 정의되어 있다. 리스너 클래스를 만들 때는 이 인터페이스를 상속(implements)하여 구현해야 한다.

안드로이드 버튼의 클릭 이벤트 핸들러는 버튼의 자식 클래스를 만들어서 구현하지 않고, OnClickListener 리스너 클래스를 만들어서 구현해야 한다.

짧게 줄이지 않고 자세히 말하면, 버튼의 클릭 이벤트 핸들러는 OnClickerListener 인터페이스를 상속(implements)하여 자식 클래스를 만들고, 그 자식 클래스에서 onClick 이벤트 핸들러 메소드를 재정의하여 구현해야 한다.

리스너로 구현한 경우에는 메소드 구현으로 끝나는 것이 아니고, 구현한 리스너 클래스의 객체를 하나 생성하여 버튼에 등록까지 해야 한다. 이렇게 리스너 객체를 등록하는 메소드 이름의 관례는 **setOn이벤트Listener** 이다.

예를 들어 click 이벤트의 경우

이벤트 핸들러 메소드 이름 onClick

리스너 인터페이스 이름 OnClickListener

리스너 등록 메소드 이름 setOnClickListener

(줄21~30)의 코드를 다음과 같이 구현해도 된다. 동일한 코드이다.

b.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

EditText e = (EditText)findViewById(R.id.editText1);

CharSequence s = e.getText();

TextView t = (TextView)findViewById(R.id.textView1);

t.setText(s);

}

});

위 코드가 동일한 이유는 아래 코드들이 동일한 이유와 같다.

String[] arr = new String[] {

"a", "b", "c", "d"

};

obj.setData(arr);

String 객체의 배열을 생성하고, 그 배열에 대한 참조를 지역 변수 arr에 저장하였다.

obj 객체의 setData 메소드를 호출하면서, 그 배열에 대한 참조를 파라미터로 전달하였다.

이렇게 두 문장으로 구현된 코드를 아래의 코드에서는 한 문장으로 결합하였다.

결과는 동일하다.

obj.setData(new String[] {

"a", "b", "c", "d"

});

(줄21~30)는 간단히 말해서 OnClickListener 객체를 생성하여 버튼에 등록하는 문장이다.

이 문장은 액티비티의 onCreate 메소드에 들어있으므로, 액티비티의 onCreate 메소드가 호출될 때 실행된다.

액티비티의 onCreate 메소드는 액티비티가 생성되자마자 즉시 호출된다.

(줄23)의 onClick 이벤트 핸들러는 리스너 객체의 메소드이다. 줄21~29는 리스너 객체를 생성해서 버튼에 등록만 했을뿐이지, 리스너 객체의 onClick 메소드를 호출하지 않았다.

리스너 객체의 onClick 메소드는, 리스너 객체가 등록되어 있는 버튼이 클릭되었을때 자동으로 호출된다.

(줄24) 액티비티의 화면의 내용에 해당하는 뷰 객체들 중에서, id가 R.id.editText1인 객체를 찾아서, 그 객체에 대한 참조를 리턴한다. 리턴된 참조를 EditText 타입의 지역 변수에 저장한다.

액티비티의 화면의 내용에 해당하는 뷰 객체들은 줄18의 메소드 호출에서 생성되었다.

(줄25) EditText는 문자열을 입력하기 위한 텍스트 박스이다. getText 메소드를 호출하여 EditText에 입력된 문자열을 얻는다. getText 메소드가 리턴하는 것은 CharSequence 타입의 객체이다.

Java 언어에서 문자열 타입의 클래스들의 공통 부모 클래스에 해당하는 것이 CharSequence 인터페이스이다.

(줄26) 액티비티의 화면의 내용에 해당하는 뷰 객체들 중에서, id가 R.id.textView1인 객체를 찾아서, 그 객체에 대한 참조를 리턴한다. 리턴된 참조를 TextView 타입의 지역 변수에 저장한다.

액티비티의 화면의 내용에 해당하는 뷰 객체들은 줄18의 메소드 호출에서 생성되었다.

(줄27) TextView 클래스의 setText 메소드를 호출하여, 화면에 표시될 문자열을 전달한다.

(줄24~27) 요약하자면, EditText에 입력된 문자열을 꺼내서 TextView에 표시될 문자열로 전달한다.

이것은 버튼이 클릭되면 실행된다.

(줄16~31) 액티비티의 onCreate 메소드는 액티비티가 시작되자 호출되어 실행된다.

이 정도 코드를 실행하는 것은 몇 밀리초도 걸리지 않으니 onCreate 메소드는 금방 리턴된다.

onCreate 메소드가 리턴된 후 이 앱은 특별히 할 일이 없으니 대기 상태가 된다.

사용자가 버튼을 클릭하면 그때 줄22의 onClick 메소드가 호출된다.

onClick 메소드도 아주 간단한 코드라서 금방 실행을 마치고 리턴할 것이다.