첫번째 예제

2017-08-29

이승진

**학습목표**

안드로이드 기초 개념 소개

간단한 안드로이드 앱 만들기

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

anonymous inner class 문법 이해하기

**목차**

[1. 안드로이드 소개 2](#_Toc491775509)

[1) 안드로이드 시스템의 계층 구조 2](#_Toc491775510)

[2) API (Application Programming Interface) 2](#_Toc491775511)

[3) 안드로이드 애플리케이션의 주요 구성요소 3](#_Toc491775512)

[2. 첫번째 예제 (E01Button) 4](#_Toc491775513)

[1) 프로젝트 생성 4](#_Toc491775514)

[2) activity\_main.xml 수정 7](#_Toc491775515)

[3) 앱 실행 15](#_Toc491775516)

[4) lambda expression으로 자동 수정 기능 끄기 16](#_Toc491775517)

[5) MainActivity.java 수정 17](#_Toc491775518)

[3. 용어 정리 23](#_Toc491775519)

[4. 과제 24](#_Toc491775520)

# 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)

안드로이드 앱 개발을 공부한다는 것은 안드로이드 API를 공부한다는 말이다.

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 애플리케이션의 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.

### Service

화면에 보이지 않고 백그라운드에서 실행되는 서비스 객체이다.

### Context

Activity 클래스와 Service 클래스의 부모 클래스이다.

정확하게 말하자면, Context는 부모 클래스가 아니고 부모에 해당하는 interface 이다.

### Content Provider

이 앱의 데이터를 관리하고, 다른 앱에 데이터를 제공하는 일을 하는 객체이다.

앱에서 필요한 데이터를 가져오는 방법은 다음과 같다.

- SQLite 데이터베이스에서 직접 읽어오기

- 인터넷을 통해서 서버로부터 받아오기

- 컨텐트 프로바이더를 통하여 데이터를 받아오기

컨텐츠 프로바이더를 통해서 데이터를 받아와야 하는 대표적인 사례가 전화번호 목록이다.

전화번호 목록을 여러 앱이 직접 마음대로 읽고 쓰고 하는 것은 바람직하지 않다.

어떤 통제가 필요하다.

전화번호 컨텐츠 프로바이더에게 요청해서 전화번호 목록을 받아가야 한다.

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

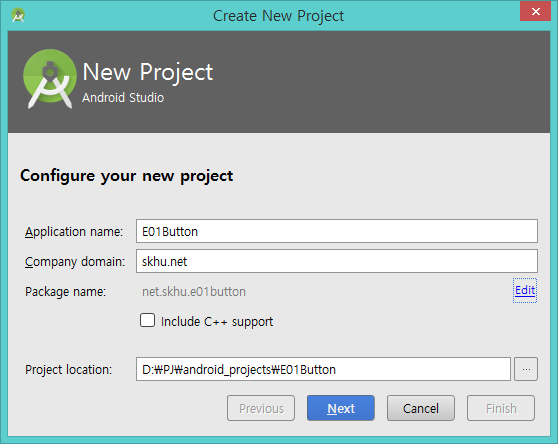
- 주소 편집 화면을 구현한 activity 객체

- 다른 앱에 주소 데이터를 제공하기 위한 content provider 객체

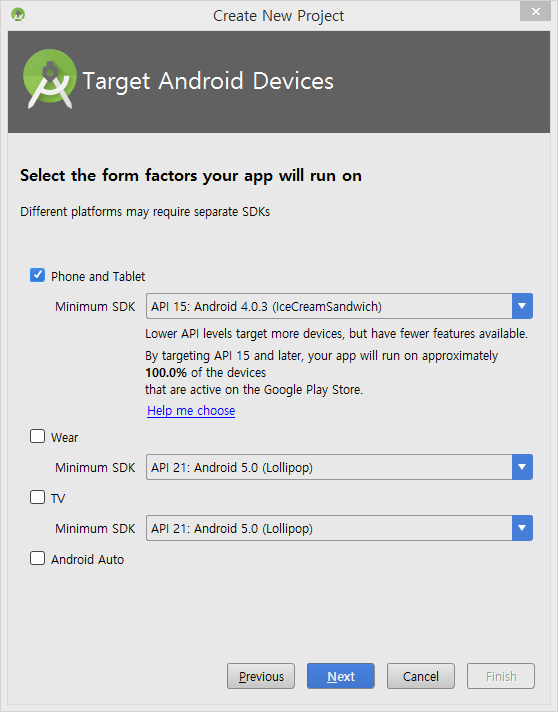
# 첫번째 예제 (E01Button)

## 프로젝트 생성

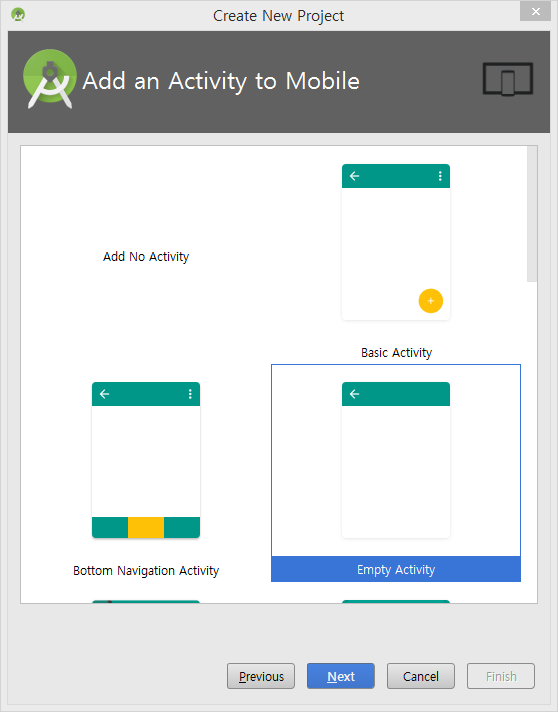
메뉴: File – New – New Project



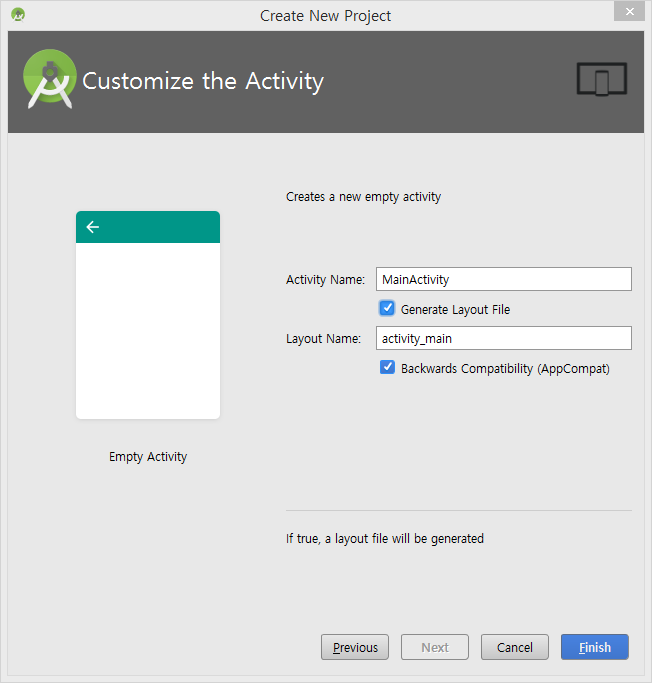
위와 같이 입력하고 Next를 계속 눌러서 진행하자.



Next 버튼 클릭



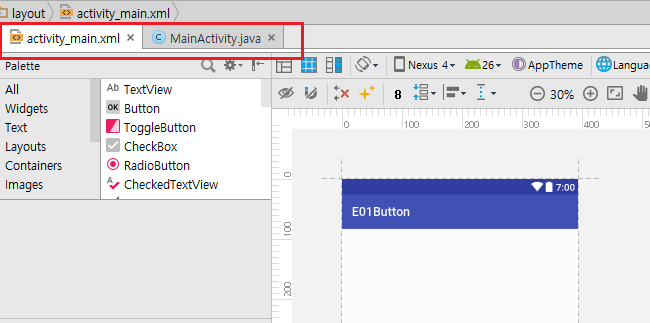
Next 버튼 클릭



Finish 버튼 클릭

Finish 버튼을 클릭하고 한참을 기다리면

프로젝트가 생성되고 다음과 같이 편집창이 열리게 된다.



편집창 위의 탭에 파일명이 보인다.

액티비티 소스 파일 두 개로 자동으로 생성되었고,

이 두 파일의 편집창이 열려있는 상태이다.

자동으로 생성된 파일은 MainActivity.java와 activity\_main.xml 이다.

액티비티는 이렇게 java 파일 하나와 xml 파일 하나로 구현된다.

MainActivity.java 파일은 액티비티의 소스 코드 파일이고

ativity\_main.xml 파일은 액티비티의 화면 형태를 저장하는 파일이다.

## activity\_main.xml 수정

텍스트 박스, 체크 박스, 버튼 등 화면의 구성 요소를 안드로이드에서는 뷰(view) 객체라고 부른다.

이 뷰 객체들을 화면에 배치하는 방식을 레이아웃(layout)이라고 부른다.

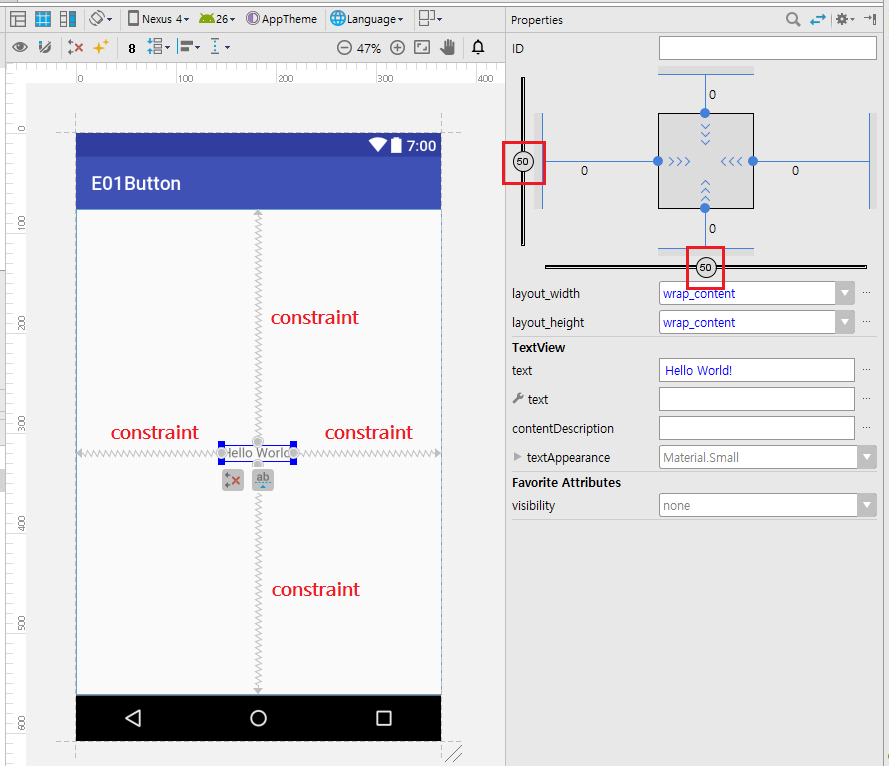
activity\_main.xml 파일의 내용은 다음과 같다.

- 앱의 첫 화면에 포함될 뷰 객체들의 목록

- 크기, 폰트, 배경색 같은 뷰 객체들의 속성

- 뷰 객체들의 배치 레이아웃

### ConstraintLayout



화면 가운데 "Hello World" 문자열을 표시하는 것은 TextView 객체이다.

이 TextView 객체 주위의 용수철 모양의 화살표 4개는 constraint 이다.

이 constraint에 의해서 TextView의 위치가 결정된다.

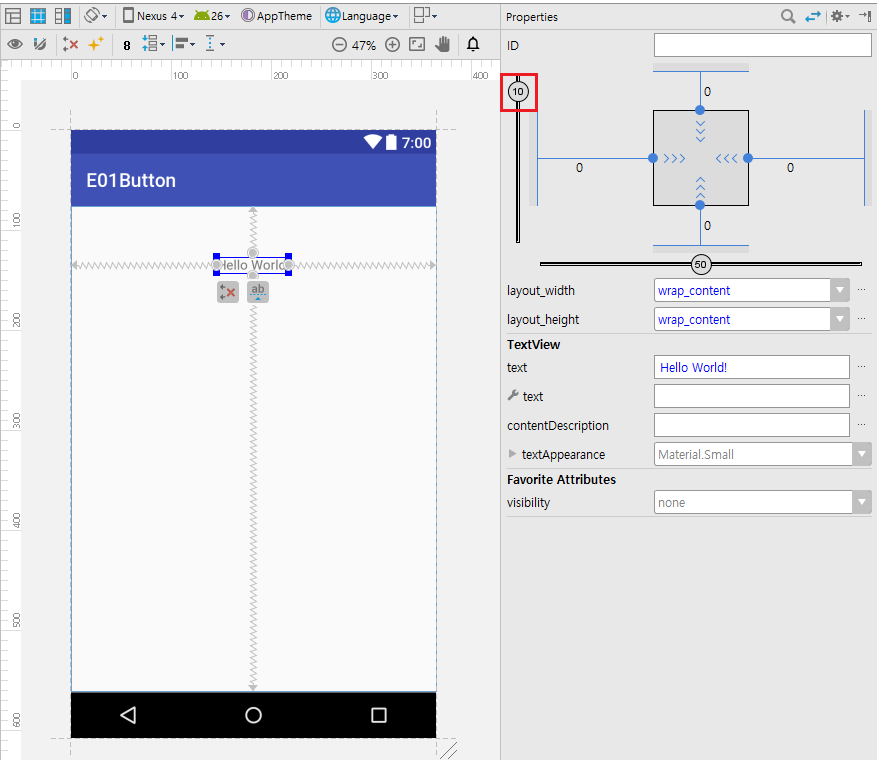
TextView의 위치는

화면 위쪽 끝과 아래쪽 끝 사이의 50 : 50 인 지점

화면 왼쪽 끝과 아래쪽 끝 사이의 50 : 50 인 지점이다.

TextView를 마우스로 드래그 하여 위치를 옮겨보자.

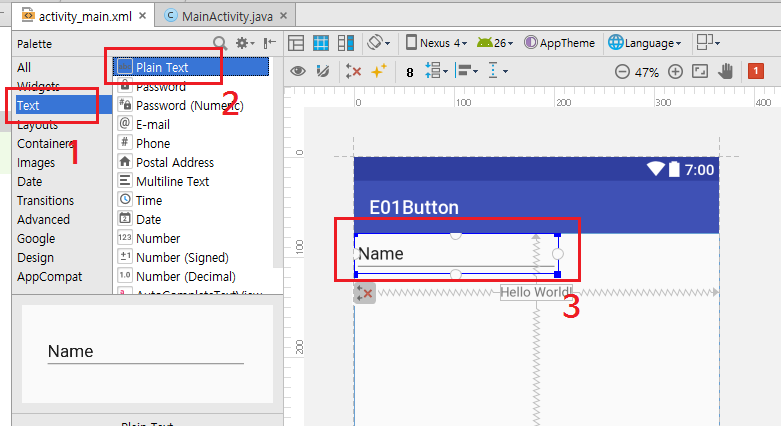
화면에 표시된 50 숫자가 변경될 것이다.



화면 위쪽 끝과 아래쪽 끝 사이의 10 : 90 인 지점에 TextView가 위치하게 된다.

이런 방식으로 위치를 결정하는 것을 ConstraintLayout 이라고 부른다.

### EditText 추가하기



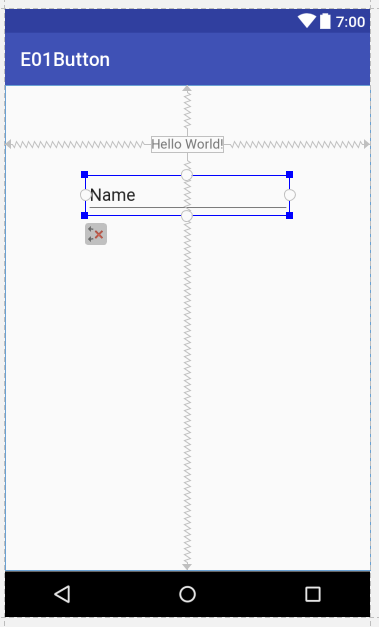
위 화면에서 1을 클릭하면, 화면에 텍스트를 입력하기 위한 객체들 목록이 오른쪽에 나타난다.

화면에 텍스트를 입력하기 위한 객체가 EditText 이다.

2를 드래그하여 화면에 드롭하자.

화면에 EditText 객체가 추가된다.

이 EditText를 드래그하여 위치를 옮겨보자.



이 EditText에는 constraint 연결선이 붙어있지 않다.

이 EditText의 위치는 x, y 값 얼마 얼마로 고정된 위치이다.

스마트폰 화면 크기와 무관하게 x, y 좌표가 고정되었다.

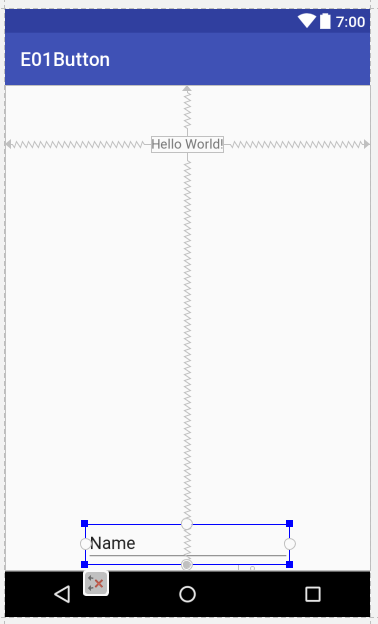
이 방식은 별로 바람직하지 않다.

constraint 연결선으로 위치를 지정하면

스마트폰 화면 크기에 따라 적절하게 위치가 조정되기 때문에 바람직하다.

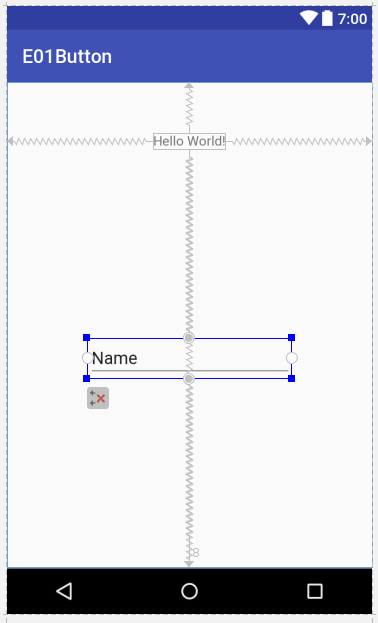
EditText 경계에 표시된 작은 동그라미가 constraint 연결선을 만들기 위한 핸들이다.

4개의 동그라미 중에서 아래쪽 동그라미를 드래그 드롭하여 화면 아래쪽 끝에 연결하자.



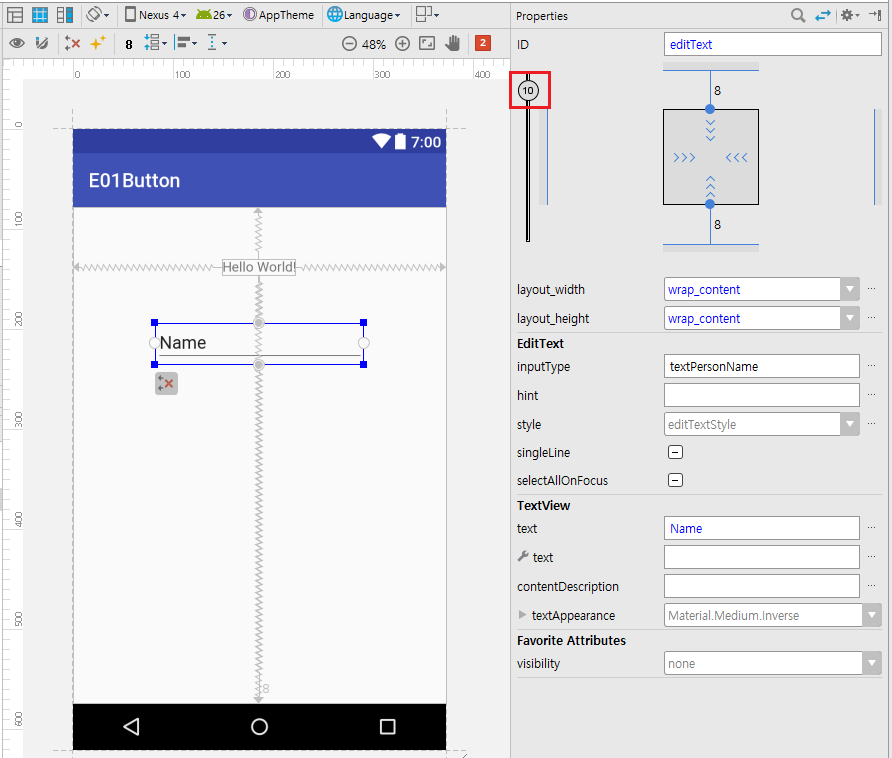
화면 아래쪽에만 연결되었기 때문에, EditText 위치도 화면 아래에 붙게된다.

위쪽 동그라미를 드래그 드롭하여 "Hello World" 기 표시된 TextView의 아래에 연결하자.



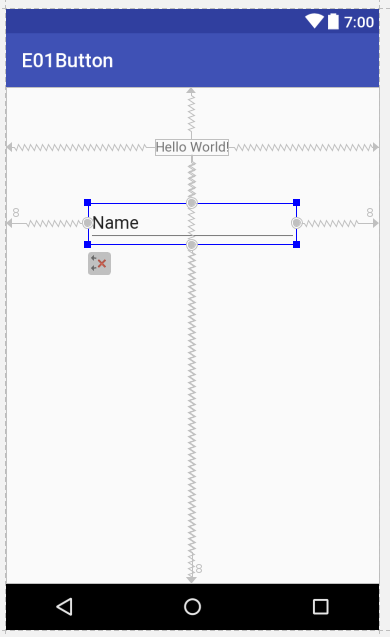
TextView와 화면 아래 끝 사이의 50 : 50 인 곳에 EditText가 위치하게 된다.

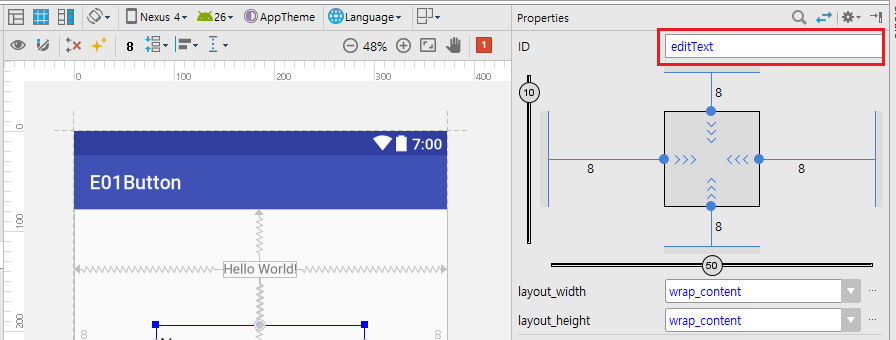
EditText를 약간 위로 옮기자.



TextView와 화면 아래 끝 사이의 10 : 90 인 곳에 EditText가 위치하게 된다.

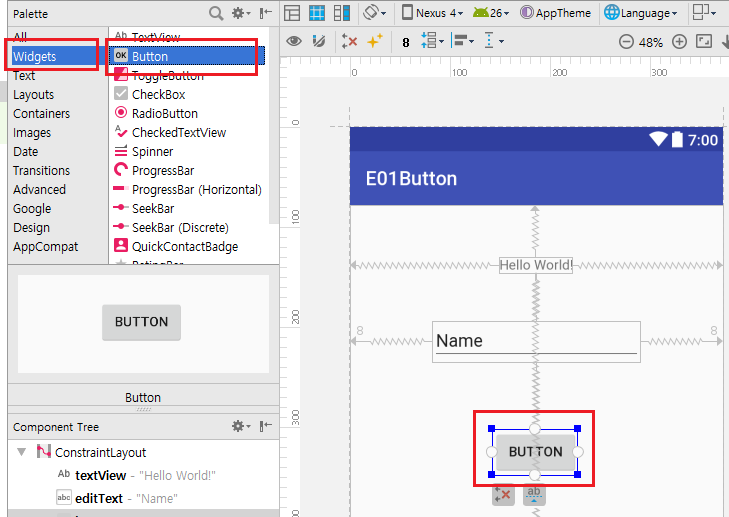
EditText의 좌우 contstraint 연결선도 연결하자.





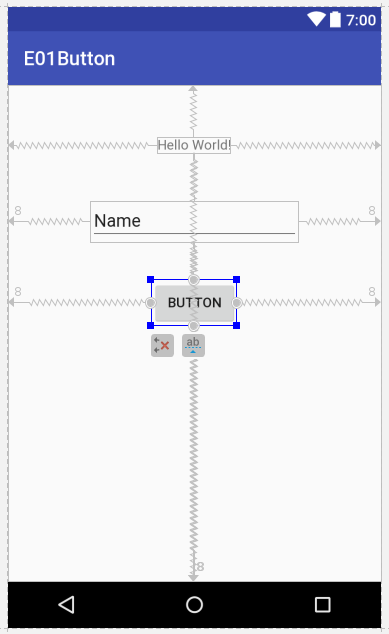
EditText의 ID를 확인하자.

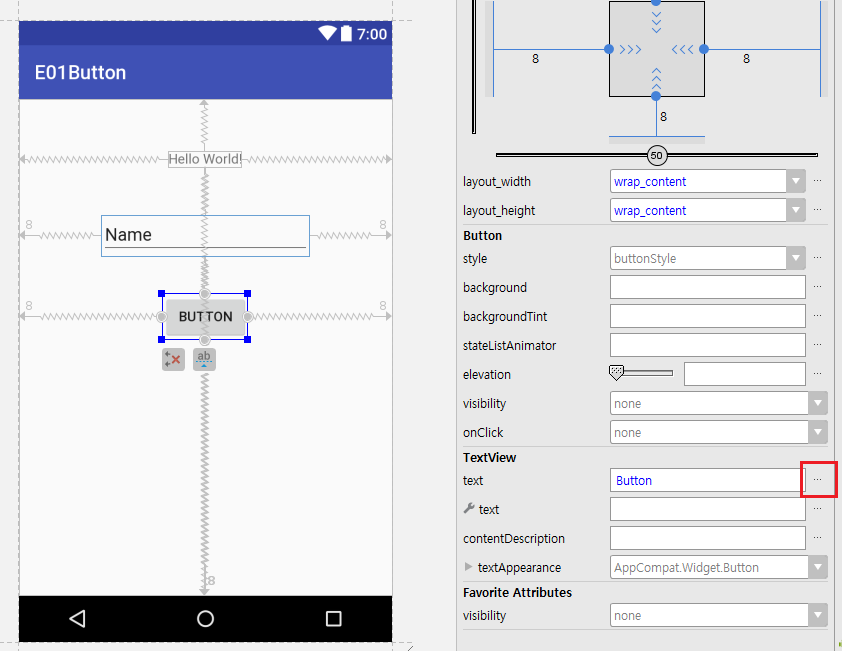
### Button 추가하기



Button 객체를 추가하자.

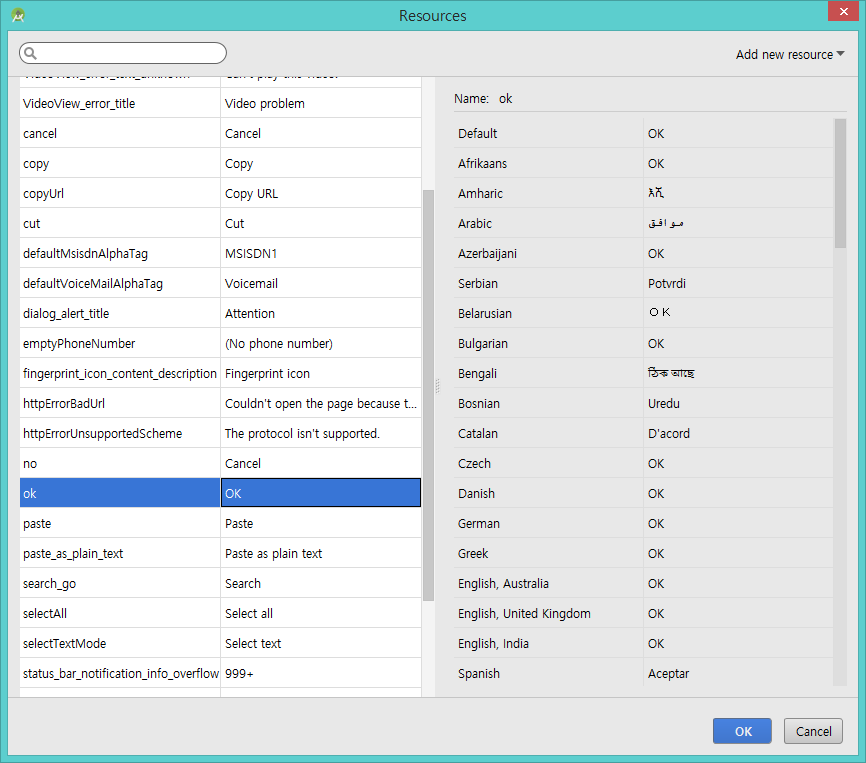
EditText와 동일한 방법으로 constraint 연결선을 연결하자.





버튼의 텍스트를 "Ok"로 변경하자.

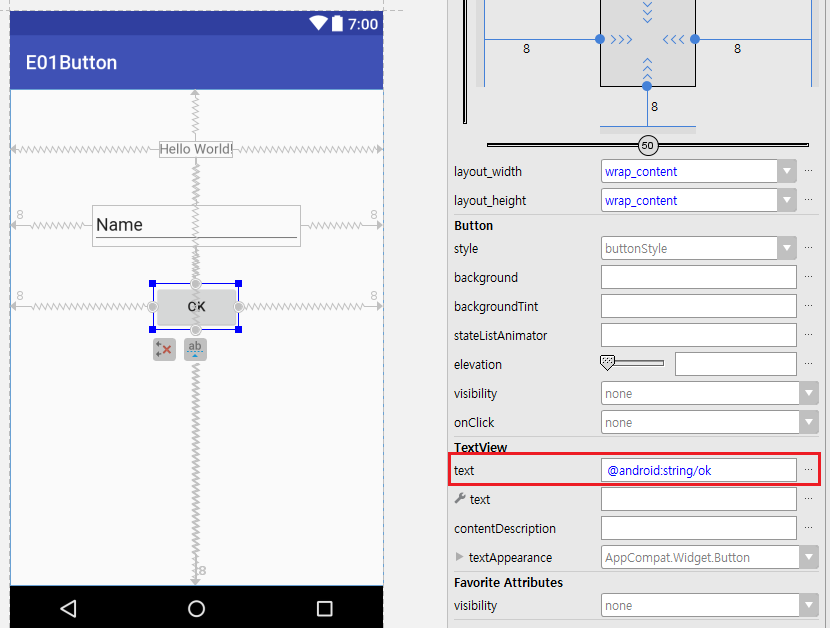
화면 오른쪽 Properties 창에서 text 속성 옆의 ... 부분을 클릭하자.



문자열 리소스 창이다.

이 창에서 Ok 항목을 선택하고

Ok 버튼을 클릭하자.



버튼의 text 속성값이 변경되었다.

|  |
| --- |
| @android:string/ok |

이것은 문자열 리소스의 id 이다.

### 문자열 리소스

xml 파일에 문자열을 등록하고 사용하는 것이 문자열 리소스이다.

문자열을 Java 소스코드에 입력하지 않기 위해서 xml 파일을 사용한다.

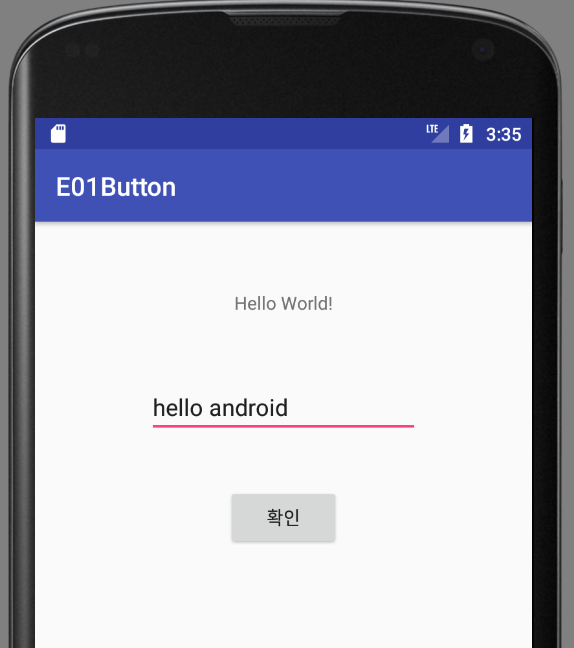
앱에 표시된 문자열을 영어, 프랑스어, 한글로 바꿀 수 있으려면,

문자열을 Java 소스코드에 입력하면 안된다.

xml 문자열 리소스 파일에 문자열을 입력해야 여러 나라 언어로 변경하기 쉽다.

## 앱 실행

메뉴: Run - Run App



EditText에 아무 텍스트나 입력해 보자.

그리고 Ok 버튼을 클릭해 보자.

버튼을 클릭해도 아무런 일이 일어나지 않을 것이다.

버튼이 클릭되었을 때 일어나는 일을 아직 구현하지 않았기 때문이다.

## lambda expression으로 자동 수정 기능 끄기

Java 언어에 anonymous inner class 문법이 있다.

Android 앱을 구현할 때 이 문법을 자주 사용하게 된다.

그런데 Java 버전 8에서 lambda expression 문법이 추가되었다.

구현할 메소드가 하나인 anonymous inner class인 경우에, lambda expression 문법으로 구현하면 훨씬 더 간결하니 편리하다.

Java 언어가 Java 8에서 함수형 프로그래밍 패러다임(functional programming paradigm)을 지원하기 위한 문법이 추가되면서, 함수형 프로그래밍 언어의 대표적인 문법인 lambda expression 문법도 Java 8에 추가되었다.

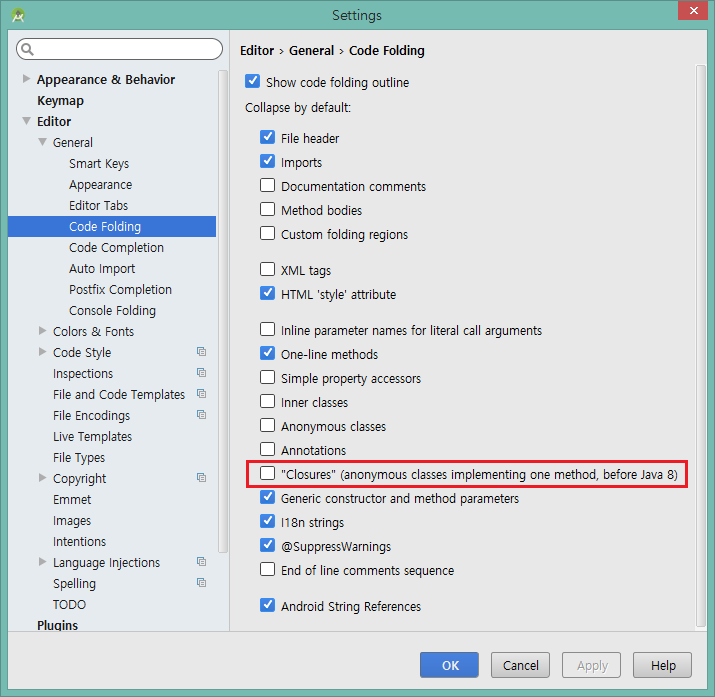
절차적 프로그래밍 패러다임과 비교하여 함수형 프로그래밍 패터다임의 장점은, 병렬성 구현이 무척 쉽고, 복잡한 자료 처리 로직을 아주 간단하게 구현할 수 있다는 점이다. 함수형 프로그래밍 패러다임의 단점은, 익숙한 C, C++, Java 언어와 문법이 많이 달라서 익숙해지기까지 시간이 걸린다는 점이다.

Android Studio는 anonymous inner class 문법으로 구현된 소스 코드를 자동으로 lambda expression 문법으로 수정해준다. lambda expression 문법을 잘 안다면 이 자동 수정 기능이 편리하고 좋지만, lambda expression 문법을 모른다면 오히려 혼란스러울 것이다.

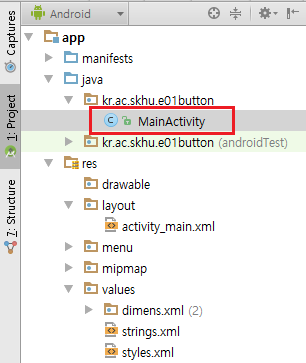
먼저 anonymous inner class 문법에 익숙해야 하고, 그 다음에 lambda expression 문법을 공부하자.

lambda expression 문법을 공부하기 전까지, Android Studio의 자동 수정 기능을 끄자.

메뉴: File - Settings - Editor - General - Code Folding - "Closures" (anonymous classes implementing one method, before Java 8) - 체크 끄기



## 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 | **package** net.skhu.e01button;  **import** android.support.v7.app.AppCompatActivity;  **import** android.os.Bundle;  **import** android.view.View;  **import** android.widget.Button;  **import** android.widget.EditText;  **import** android.widget.TextView;  **public class** MainActivity **extends** AppCompatActivity {  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_main***);  Button b = (Button)findViewById(R.id.***button***);  View.OnClickListener listenerObj = **new** View.OnClickListener() {  @Override  **public void** onClick(View v) {  EditText e = (EditText)findViewById(R.id.***editText***);  CharSequence s = e.getText();  TextView t = (TextView)findViewById(R.id.***textView***);  t.setText(s);  }  };  b.setOnClickListener(listenerObj);  }  } |

위와 같이 수정하자.

위 코드의 대부분은 자동으로 생성된 코드이고, 새로 추가된 부분은 (줄5~8), (줄17~27)이다.

|  |
| --- |
| **public class** MainActivity **extends** AppCompatActivity { |

안드로이드 앱의 화면을 담당하는 액티비티 클래스이다.

이 클래스의 부모 클래스는 AppCompatActivity 클래스이다.

모든 액티비티 클래스의 부모 클래스는 Activity 클래스이다.

|  |
| --- |
| @Override |

부모 클래스의 메소드를 재정의(override)하는 경우, 그 메소드의 앞에 붙여주는 어노테이션(annotation)이다. 부모 클래스인 Activity 클래스에 onCreate 메소드가 이미 정의되어 있다. 이 메소드를 자식 클래스인 MainActivity 클래스에서 재정의 하고 있다.

|  |
| --- |
| **protected void** onCreate(Bundle savedInstanceState) { |

안드로이드 앱의 화면이 시작될 때, 그 화면에 해당하는 액티비티 클래스의 onCreate 메소드가 저절로 호출된다. 따라서 안드로이드 앱의 화면이 처음 시작될 때 해야할 일을 onCreate 메소드을 재정의하여 구현해야 한다.

|  |
| --- |
| **super**.onCreate(savedInstanceState); |

부모 클래스의 onCreate 메소드를 호출한다.

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

onCreate 메소드를 구현할 때, 언제나 첫 줄은 super.onCreate(savedInstanceState); 이어야 한다.

|  |
| --- |
| setContentView(R.layout.***activity\_main***); |

R.layout.activity\_main은 int 형 상수이다.

앞에서 화면 레이아웃을 res/layout/activity\_main.xml 파일에 작성하였다.

이 리소스 파일을 Java 소스코드에서 사용하려면 id가 필요하다.

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

Java 소스 코드에서 레이아웃 리소스 파일을 지정할 때, 경로명을 사용하지 않고, id를 사용한다.

setContentView(R.layout.***activity\_main***);

setContentView 메소드는 Activity 클래스의 메소드이다.

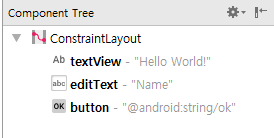
이 메소드의 파라미터는 레이아웃 리소스 파일의 id 이다.

이 메소드는 레이아웃 리소스 파일의 내용대로 뷰(view) 객체들을 생성해서 액티비티의 화면에 채우는 일을 한다.

activity\_main.xml 파일의 내용대로 TextView, EditText, Button 객체가 자동으로 만들어진다.

화면의 구성 요소인 TextView, EditText, Button 등의 객체를 안드로이드에서는 뷰(view) 객체라고 부른다.

Activity 클래스의 setContentView 메소드 이름은, 화면의 내용(content)에 해당하는 뷰(view) 객체를 자동으로 생성해서, 화면에 채우라(set)는 뜻이다.



앞에서 작성한 activity\_main.xml 파일의 구성 요소 객체들은 위와 같다.

"Hello world!" 문자열이 채워진 TextView 객체와,

"Name" 문자열이 채워진 EditText 객체와

@android:string/ok 문자열 리소스가 표시된 Button 객체가

setContentView(R.layout.activity\_main) 메소드 호출에 의해서 자동으로 생성된다.

생성된 이 객체들을 Java 코드에서 사용하려면, id가 필요하다.

이 객체들의 id는 각각 R.id.textView, R.id.editText, R.id.button 이다.

|  |
| --- |
| Button b = (Button)findViewById(R.id.***button***); |

findViewById 메소드는 Activity 클래스의 메소드이다.

setContentView 메소드에 의해서 자동으로 생성된 객체들에 대한 참조를 얻을 때

findViewById 메소드를 사용한다.

이 메소드의 파라미트는 객체의 id 이다.

줄 17에서 호출된 findViewById 메소드는 id가 R.id.button인 Button객체에 대한 참조를 리턴한다.

이 Button 객체는 줄15의 setContentView 메소드 호출에 의해서 자동으로 생성되었다.

안드로이드에서는 Button이나 EditText 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 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 같인 참조 값만 리턴된다.

Activty 클래스의 findViewById 메소드 이름은, 화면의 내용에서 뷰(view) 객체를 id로 찾아서 리턴하는 의미이다. (find view by id)

|  |
| --- |
| Button b = (Button)findViewById(R.id.***button***); |

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 주소인 참조 값만 저장된다.

### anonymous inner class 문법

아래 Java 코드는, 리스너 객체를 하나 만드는 것을 annonymous inner class 문법으로 구현한 것이다.

이 Java 코드가 하는 일은 다음과 같다.

new OnClickListener() {

@Override

public void onClick(View v) {

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

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

3) 그 자식 클래스의 객체를 하나 생성(new)한다.

이레 Java 코드가 하는 일은 다음과 같다.

OnClickListener listenerObj = new OnClickListener() {

@Override

public void onClick(View v) {

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

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

3) 그 자식 클래스의 객체를 하나 생성(new)한다. 이 객체를 리스너 객체라고 하자.

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 주소인 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 객체를 가르키는 참조뿐이다. 참조를 객체에 대한 주소라고 생각해도 좋다.

### 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 (줄13)의 onCreate 메소드, (줄20)의 onClick 메소드가 이벤트 핸들러이다.

액티비티 클래스의 onCreate 이벤트 핸들러 메소드는 액티비티가 생성되자마자 자동으로 호출된다.

액티비티가 생성되자마자 해야할 작업을 구현해야 한다면, 액티비티 클래스에서 onCreate 메소드를 재정의(override)해서 여기에 구현하면 된다.

안드로이드 앱을 만들 때에는, 화면에 해당하는 액티비티 클래스를 구현해야 한다.

안드로이드 SDK 라이브리리에 들어있는 Activity 클래스나 AppCompatActivity 클래스를 상속 받아서

액티비티 클래스(MainActivity 클래스)를 구현해야 한다. 그 클래스(MaingActivity 클래스)에서 이벤트 핸들러 메소드를 재정의해야 한다.

### 리스너(Listener)

액티비티 클래스의 이벤트 핸들러는 액티비티의 자식 클래스를 만들어서 여기에 구현하지만,

뷰(view) 클래스의 이벤트 핸들러는 리스너(Listener) 클래스를 만들어서 리스너 클래스에 구현한다.

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.

버튼이 클릭될 때 일어나는 일을 구현하려면,

OnClickListener 인터페이스를 상속(implements)하여 자식 리스너 클래스를 만들고, 여기에 onClick 이벤트 핸들러 메소드를 구현해야 한다.

그리고 그렇게 구현된 자식 리스너 클래스의 객체를 하나 생성(new)하여, 버튼 객체에 등록해주어야 한다.

리스너 객체를 버튼에 등록할 때는 setOnClickListener 메소드를 호출한다.

버튼이 클릭되면, 그 버튼에 등록된 리스너 객체의 onClick 이벤트 핸들러 메소드가 저절로 호출된다.

버튼이 클릭될 때 자동으로 호출되는 이벤트 핸들러 메소드 이름은 onClick 이다.

onClick 메소드를 구현할 리스너 인터페이스 이름은 OnClickListener 이다.

리스너 객체를 버튼에 등록하는 메소드 이름은 setOnClickListener 이다.

이 Java 코드가 하는 일은 다음과 같다.

OnClickListener listenerObj = new OnClickListener() {

@Override

public void onClick(View v) {

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

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

3) 그 자식 클래스의 객체를 하나 생성(new)한다. 이 객체를 리스너 객체라고 하자.

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

다음과 같이 구현해도 된다. 동일한 코드이다.

b.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

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

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"

});

|  |  |
| --- | --- |
| 18  19  20  21  22  23  24  25  26  27 | View.OnClickListener listenerObj = **new** View.OnClickListener() {  @Override  **public void** onClick(View v) {  EditText e = (EditText)findViewById(R.id.***editText***);  CharSequence s = e.getText();  TextView t = (TextView)findViewById(R.id.***textView***);  t.setText(s);  } }; b.setOnClickListener(listenerObj); |

간단히 말해서 OnClickListener 객체를 생성하여 버튼에 등록하는 Java 코드이다.

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

액티비티의 onCreate 메소드는 액티비티가 생성되자마자 즉시 호출된다.

(줄20)의 onClick 은 리스너 객체의 메소드이다. 줄18~27의 코드는 리스너 객체를 생성해서 버튼에 등록만 했을뿐이지, 리스너 객체의 onClick 메소드를 호출하지는 않았다.

리스너 객체의 onClick 메소드는, 버튼이 클릭되었을 때 자동으로 호출되고, 그때 실행된다.

줄18~27의 코드를 실행할 때, onClick 메소드는 호출되지 않는다. 나중에 버튼이 클릭되었을 때 호출될 수 있도록 버튼에 등록만 한다.

(줄21) Activity 객체의 findViewById 메소드를 호출한다. 액티비티의 내용, 즉 화면의 내용에 해당하는 뷰 객체들 중에서, id가 R.id.editText인 객체를 찾아서, 그 객체에 대한 참조를 리턴한다. 리턴된 참조를 EditText 타입의 지역 변수 e에 저장한다.

이 EditText 객체는 줄15의 메소드 호출에서 생성되었다.

(줄22) EditText는 문자열을 입력하기 위한 텍스트 박스이다. 텍스트 박스에 입력된 문자열을 얻으려면, getText 메소드를 사용한다. EditText 클래스의 getText 메소드는 텍스트 박스에 입력된 문자열을 리턴한다. getText 메소드가 리턴하는 것은 CharSequence 타입의 객체이다.

Java 언어에서 문자열에 해당하는 클래스에는 String, StringBuilder, StringBuffer 클래스가 있다.

이 문자열 클래스들의 공통 부모 클래스에 해당하는 것이 CharSequence 인터페이스이다.

(줄23) 액티비티의 내용, 즉 화면의 내용에 해당하는 뷰 객체들 중에서, id가 R.id.textView1인 객체를 찾아서, 그 객체에 대한 참조를 리턴한다. 리턴된 참조를 TextView 타입의 지역 변수에 저장한다.

이 TextView 객체는 줄15의 메소드 호출에서 생성되었다.

(줄24) TextView 클래스의 setText 메소드를 호출하여, 화면에 표시될 문자열을 전달한다.

TextView에 표시된 문자열을 바꿀 때에는 setText 메소드를 사용한다.

(줄21~24) 요약하자면, EditText에 입력된 문자열을 꺼내서 TextView에 표시될 문자열로 전달한다.

줄21~24는 onCreate 메소드가 실행될 때 실행되지 않는다.

나중에 버튼이 클릭될 때 실행된다.

(줄13~28) 액티비티의 onCreate 메소드는 액티비티가 시작되자마자 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 줄20의 onClick 메소드가 호출된다.

onClick 메소드도 아주 간단한 코드라서 금방 실행을 마치고 리턴할 것이다.

# 용어 정리

### Activity 클래스

안드로이드 앱의 화면은 Activity 클래스 객체이다.

### View 클래스

안드로이드 화면을 구성하는 Button, EditText, TextView 객체들을 뷰(view) 객체라고 부른다.

### Layout Resource XML 파일

안드로이드 앱의 화면에서 뷰(view) 객체들의 배치(layout)를 Layout Resource XML 파일에 작성한다.

### 이벤트 핸들러 메소드

버튼 클릭, 메뉴 클릭, 화면 터치와 같은 이벤트가 발생하자마자 즉시 자동으로 호출되는 메소드를 이벤트 핸들러 메소드라고 한다.

### Listener 클래스

이벤트 핸들러를 구현하기 위한 클래스이다.

### Interface (인터페이스)

Java 언어에서 interface는 class와 약간 유사한 것이다.

interface는 인스턴스 멤버 변수를 포함할 수 없다.

interface에는 보통, 메소드 본문은 없고 메소드 선언만 있는, abstract method들만 들어있다.

### OnClickListener 인터페이스

안드로이드 라이브러리에 들어있는 OnClickListener는 Java 클래스가 아니고 Java 인터페이스이다.

버튼이 클릭되었을 때 자동으로 호출되는 onClick 이벤트 핸들러 메소드를 구현하려면

OnClickListener 인터페이스를 상속(implements)하여 자식 클래스를 만들고

그 자식 클래스에 onClick 이벤트 핸들러 메소드를 구현해야 한다.

### Annonymous Inner 클래스

Java 인터페이스(예: OnClickListener 인터페이스)를 상속(implements)하여 자식 클래스를 만들고,

그 자식 클래스에서 메소드(예: onClick 메소드)를 재정의하고

그 자식 클래스의 객체를 하나 생성하는 Java 코드를 매우 간결하게 구현할 수 있는 Java 문법이다.

### Activity 클래스의 onCreate 메소드

Activity가 시작되자마자 호출되는 메소드이다.

### Activity 클래스의 setContentView 메소드

layout resource XML 파일의 내용대로 뷰 객체들을 자동으로 생성해 준다.

### Activity 클래스의 findViewById 메소드

setContentView 메소드가 자동으로 생성해준 뷰 객체에 대한 참조를 리턴해 준다.

# 과제

MainActivity 화면에 Button 하나와 EditText 두 개를 만들어라.

두 EditText에 문자열을 입력하고 버튼을 클릭하면,

EditText에 입력된 내용이 서로 바뀌도록 구현하라.