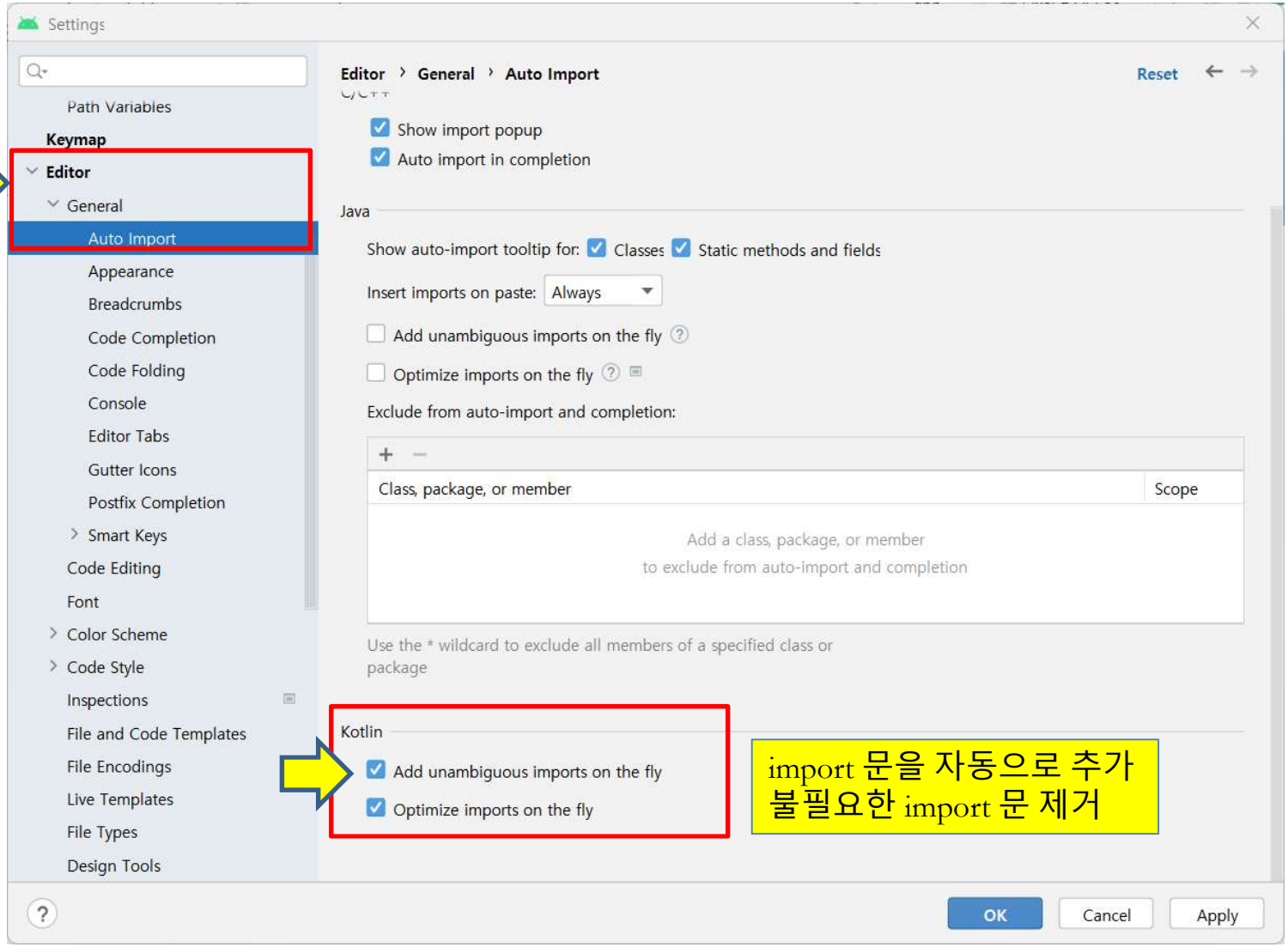
# 두 번째 애플리케이션: 응용

Mobile Software

2022 Fall

# 잠깐! Auto import 설정

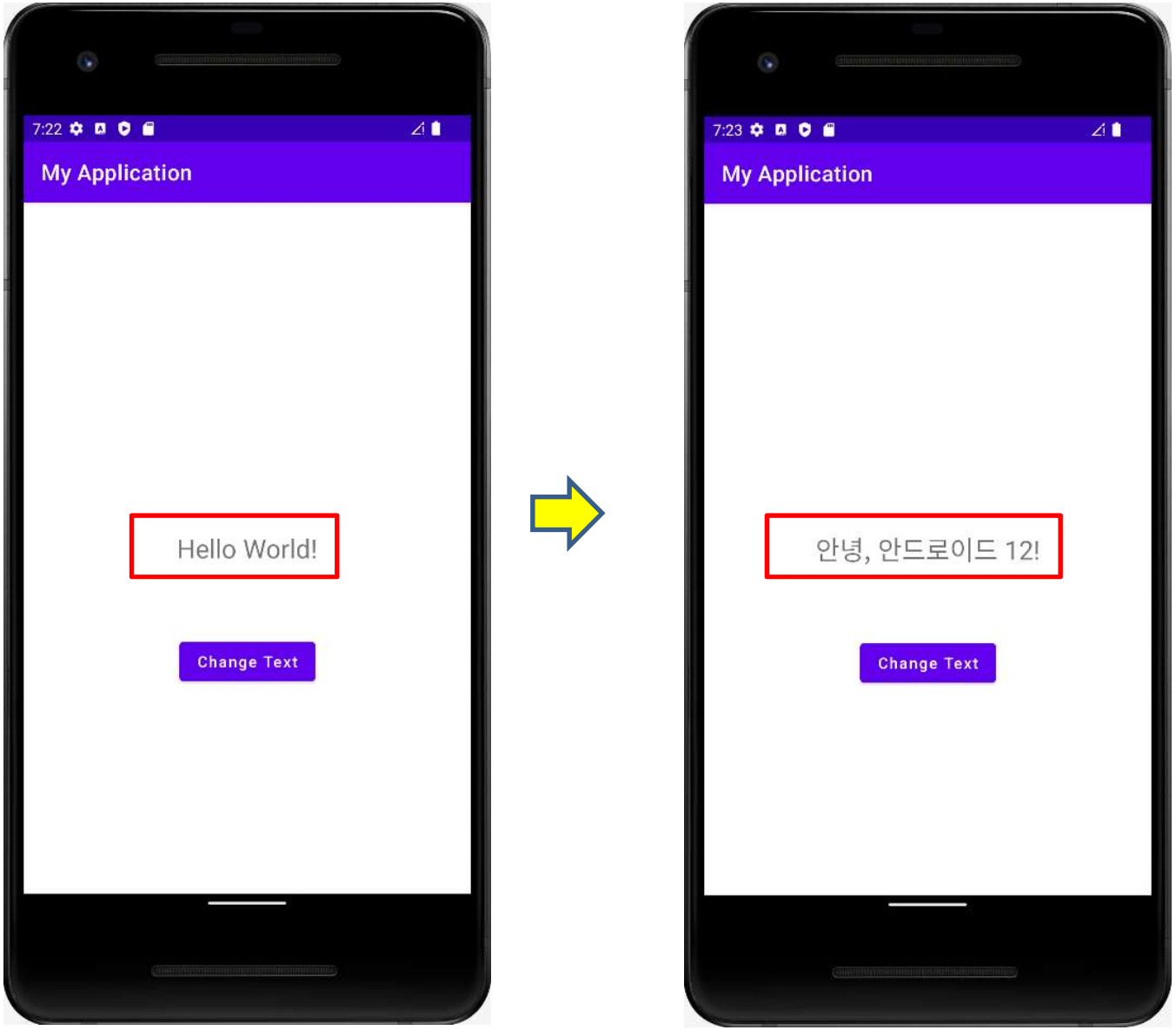
File > Settings



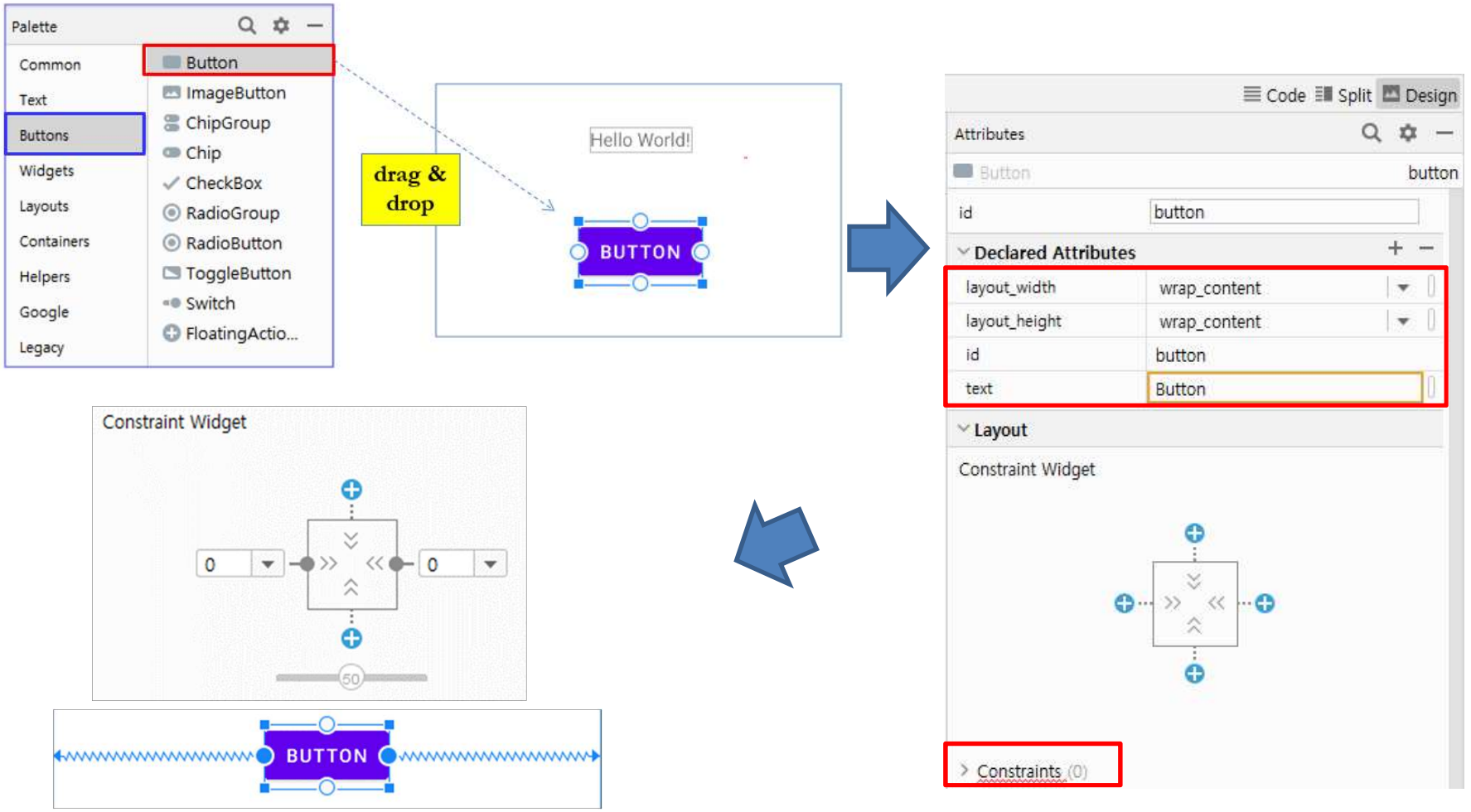
# What to do next?

- **버튼을 누르면 TextView 문자열이 바뀜**
  - **레이아웃 : Button 추가**
    - 속성 정의
      - id, text
    - 속성 함수 선언 : onBtnClicked
  - **코드**
    - 이벤트 처리 함수 구현
      - fun onBtnClicked (view: View)

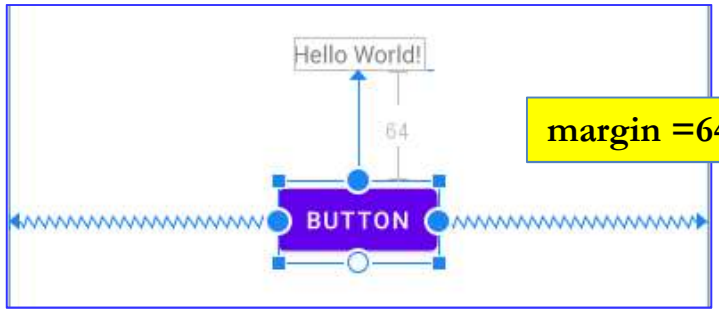
# When the button is clicked, the string is changed



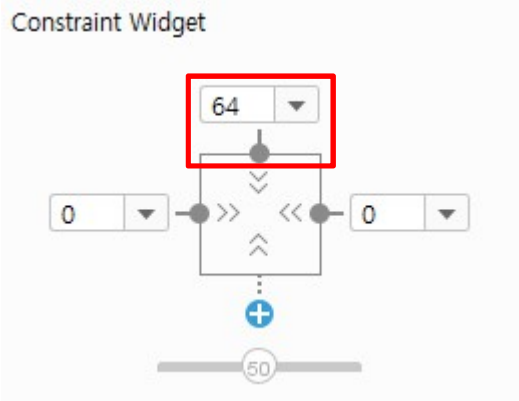
# Adding the Button (1/2)

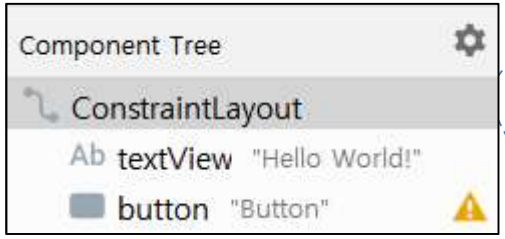


# Adding the Button (2/2)




margin =64dp



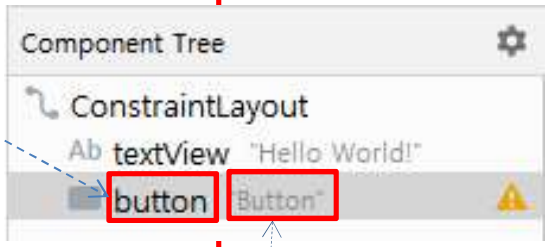




TextView 속성 추가

```
android:textSize="24sp"
```

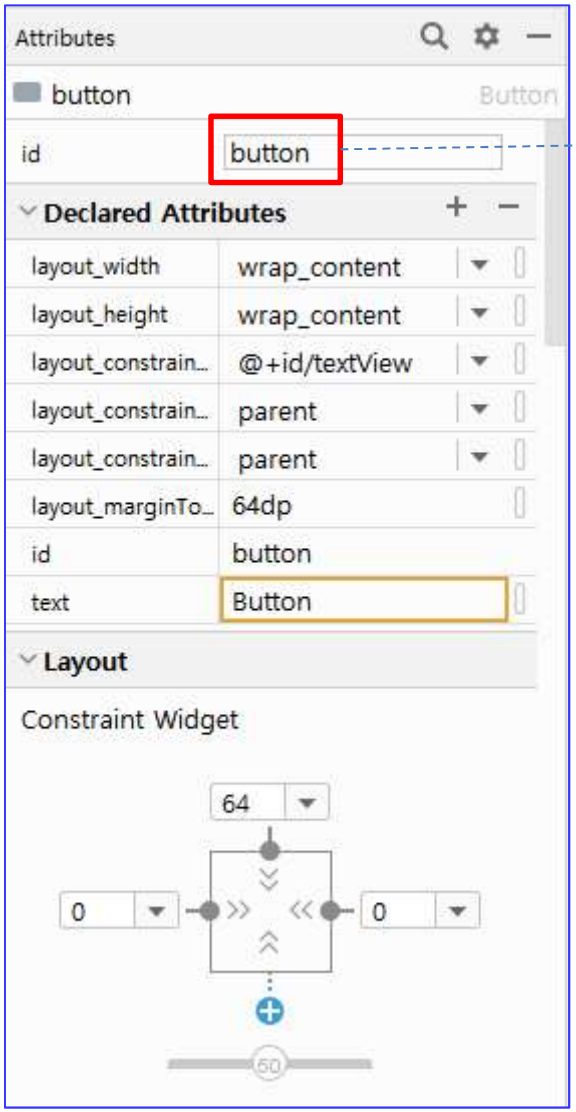

# XML Layout: activity_main.xml

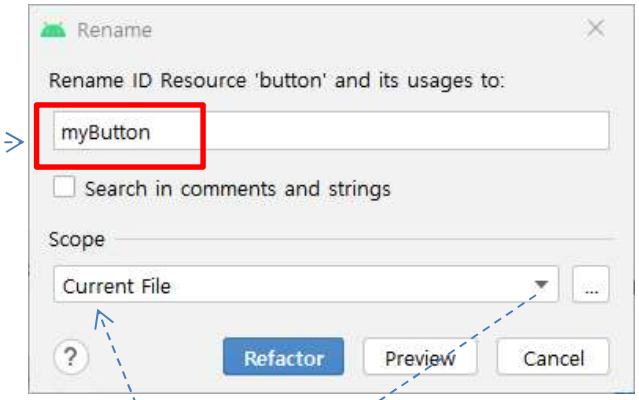
```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

    <TextView...>

    <Button
        android:id="@+id/myButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="64dp"
        android:text="Button"
        android:textAllCaps="false"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

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

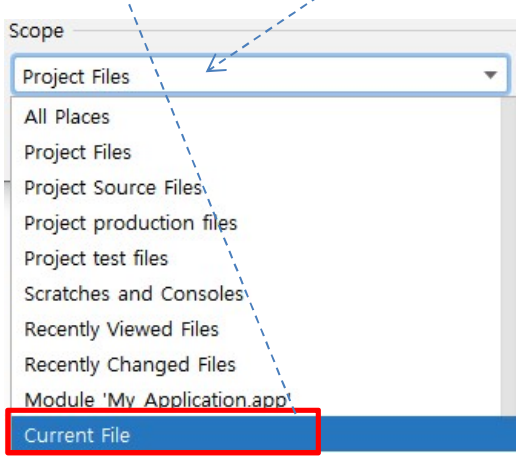
# BUTTON: Change the value of attribute id

Attributes Window

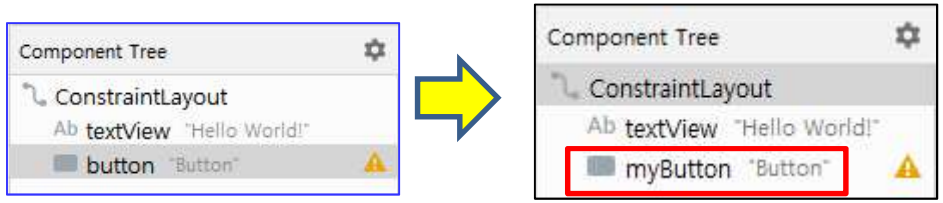



Refactor는 rename의 의미

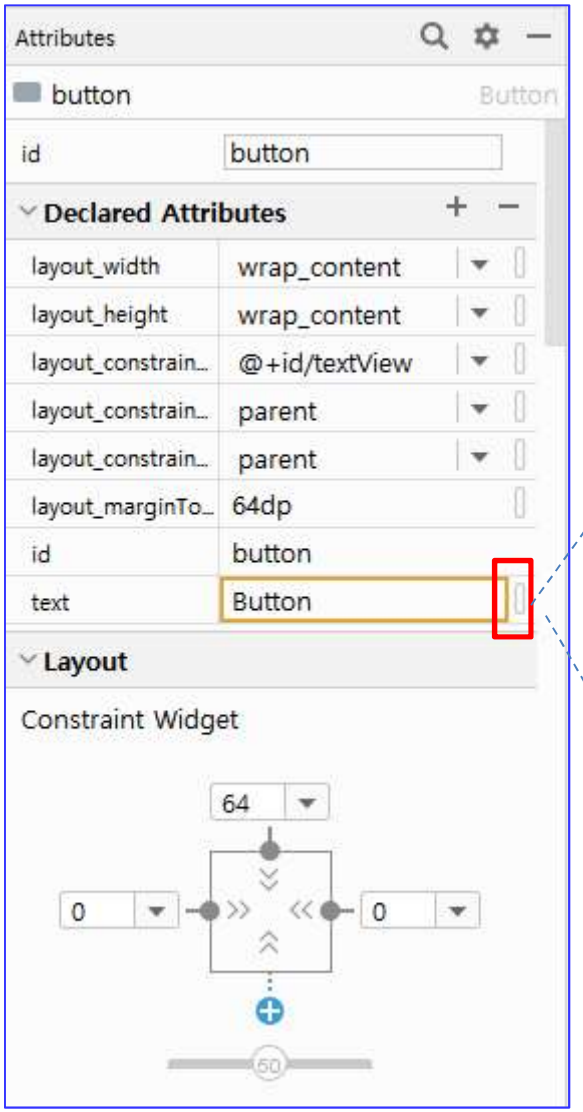


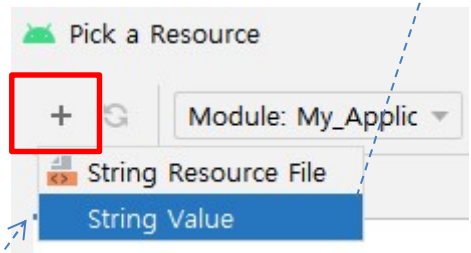

현재 파일에서만 id 속성을 변경

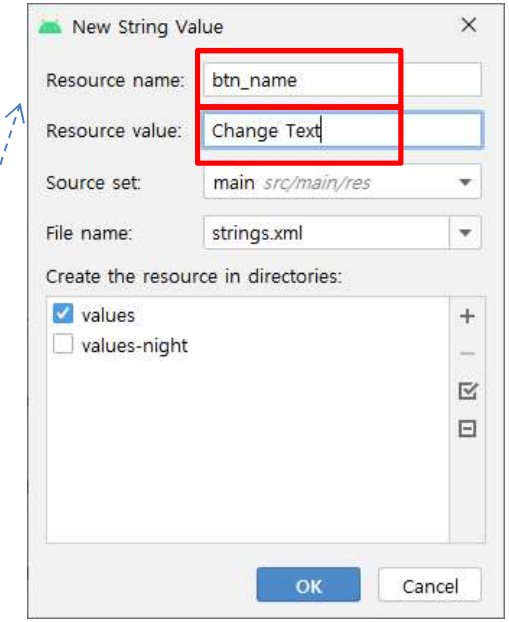


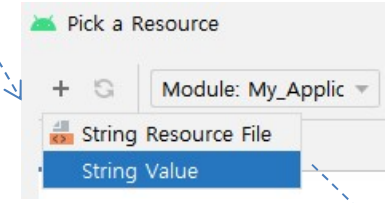
# BUTTON: Change the value of attribute text

Attributes Window

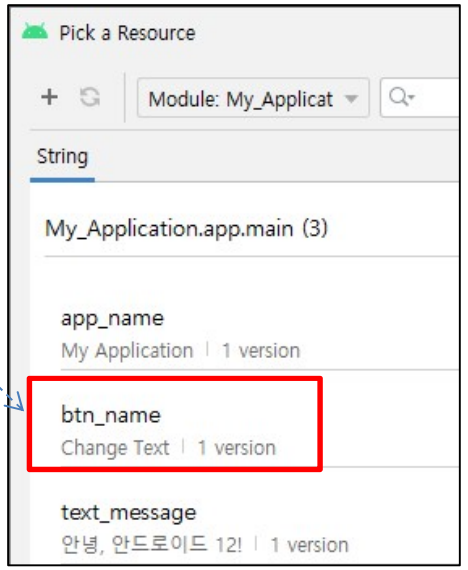


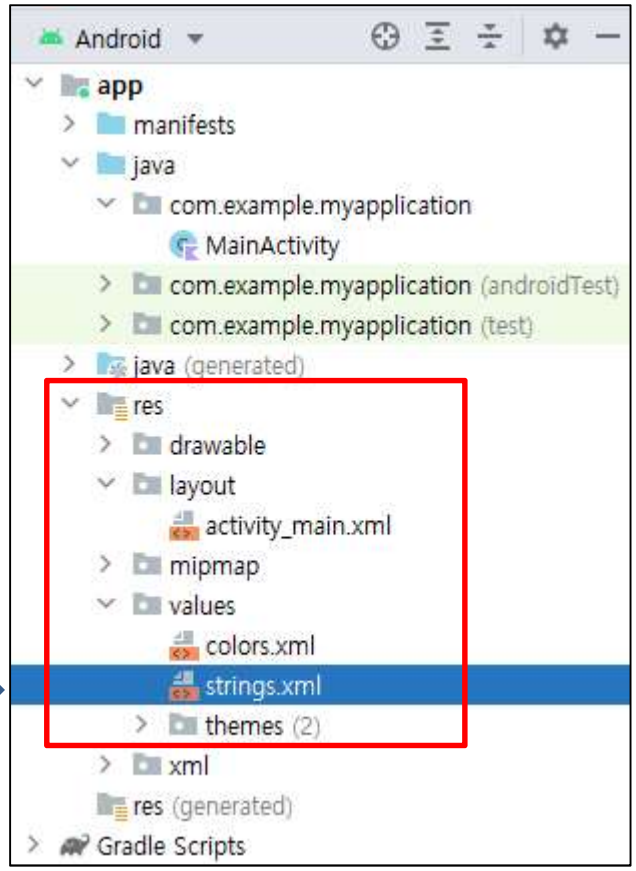



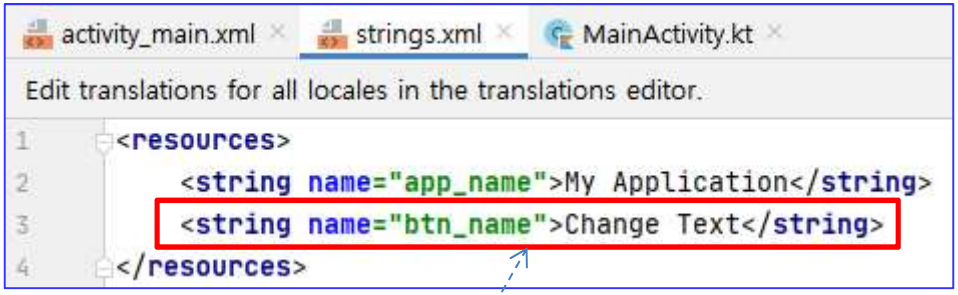




String resource에서 btn_name 을 선택



# String resource : strings.xml

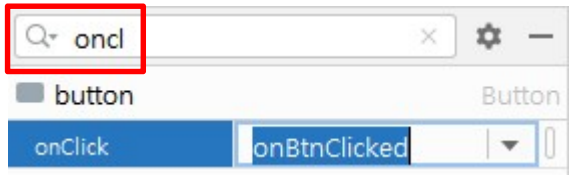

**activity_main.xml**

```
<Button
    android:id="@+id/myButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="64dp"
    android:text="@string/btn_name"
    android:textAllCaps="false"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
```

Ctrl-b 를 누르면 **strings.xml** 로 이동

# BUTTON: add two attributes

Attribute window 검색 창에 ‘ **oncl** ’ 입력




이벤트 처리 함수 설정

검색 창에 ‘ **text** ’ 입력


대소문자를 구분해 출력

```
<Button
    android:id="@+id/myButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="64dp"
    android:onClick="onBtnClicked"
    android:text="@string/btn_name"
    android:textAllCaps="false"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
```

**이벤트 처리 함수는 아직 구현하지 않았기 때문에 에러 표시**

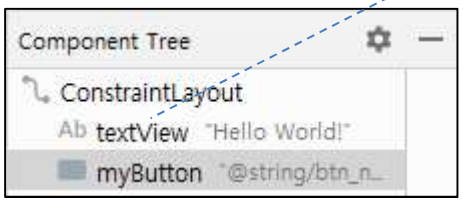
# Build the event handler function: onBtnClicked

```
package com.example.myapplication

import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun onBtnClicked(view: View) {
        val textView:TextView = findViewById(R.id.textView)
        textView.text =  "안녕, 안드로이드 12!"
    }
}
```

import 문에 추가된 클래스

MainActivity 클래스의 멤버 함수

Button을 추가할 때 onClick 속성에서 지정한 이벤트 핸들러




TextView의 id 속성 : **textView**

# 잠깐! Event handling with JavaScript

```
<Button
    android:id="@+id/myButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="64dp"
    android:onClick="onBtnClicked"
    android:text="@string/btn_name"
    android:textAllCaps="false"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
```

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun onBtnClicked(view: View) {
        val textView:TextView = findViewById(R.id.textView)
        textView.text =  "안녕, 안드로이드 12!"
    }
}
```

JavaScript 에서 사용하는 이벤트 처리 방식

Android에서 구현하려는 방식과는 **다름**

callback 메소드를 코드에서 구현할 때
**클래스의 멤버 함수**로 정의
➔ parameter는 View 객체
➔ 접근 제한: public (생략 가능)
➔ return 값 없음 : Unit (생략 가능).

# 잠깐! getter 와 setter

```kotlin
package com.example.myapplication

import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    fun onBtnClicked(view: View) {
        val textView:TextView = findViewById(R.id.textView)
        textView.text =  "안녕, 안드로이드 12!"
    }
}
```

```kotlin
textView.setText("안녕, 안드로이드 12!")
```

Kotlin에서는 getter와 setter를 자동 생성.

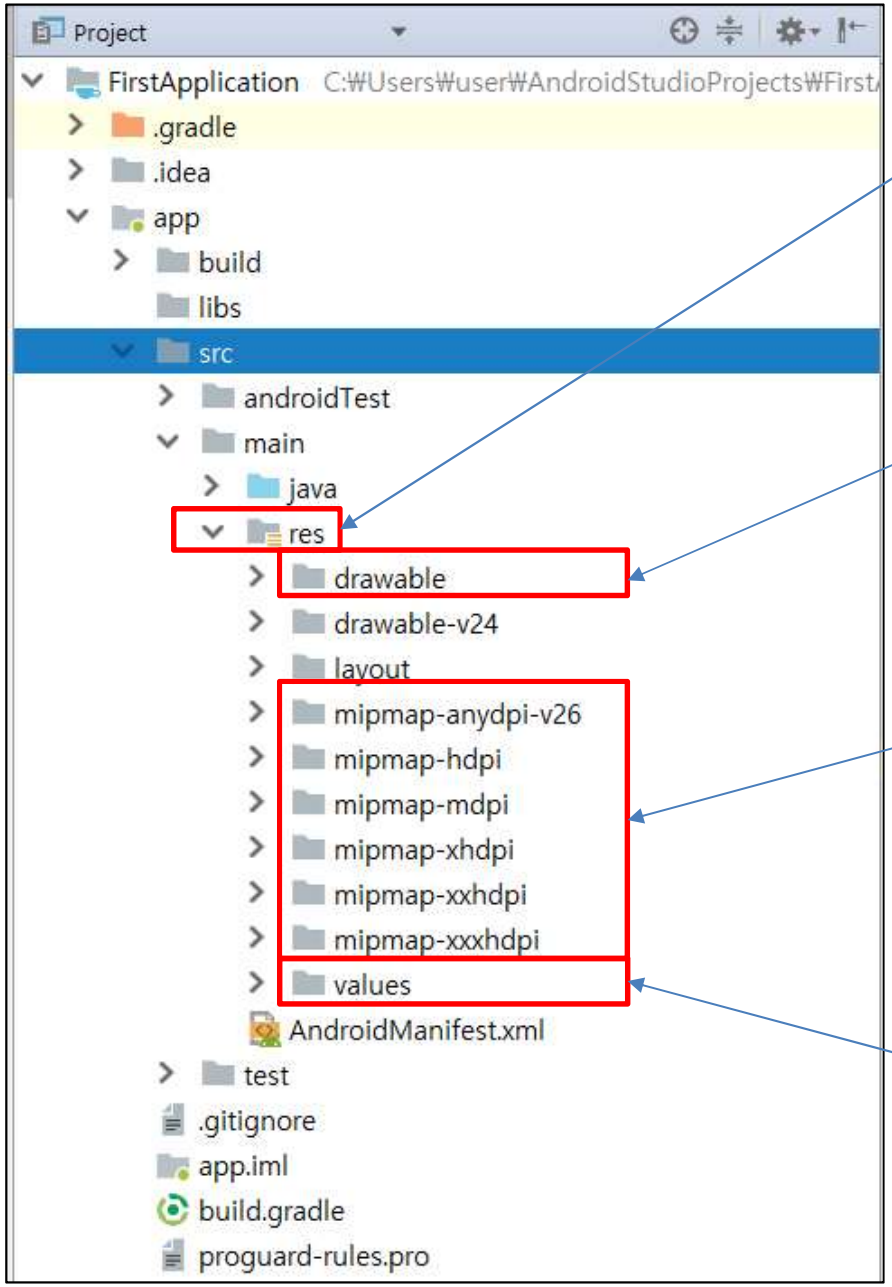
property에 직접 값을 할당.

Java에서는 **setXXX, getXXX** 과 같은 메소드를 사용.

이런 메소드를 **setter , getter** 라고 부름.

# What to do next?

- **Android Studio의 Project Window**
  - **Drawable**
  - **App. Icon**
    - Drawable, mipmap
  - **Project와 module**
  - **AndroidManifest.xml**



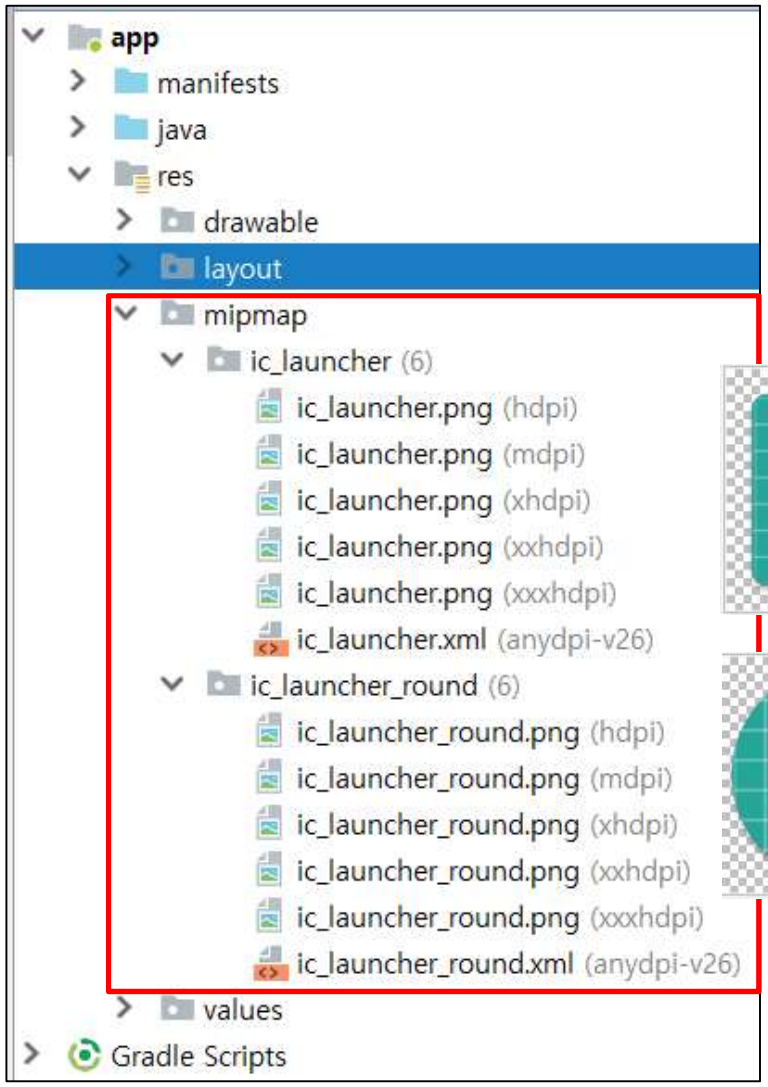

**res**
- Contains all UI resources
- Layouts, images, audio files, etc.

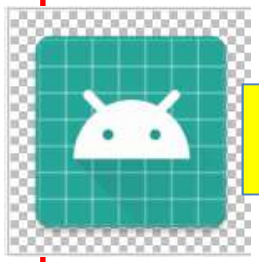
**res/drawable**
- Image assets
- Vector assets

**res/mipmap**
- App launcher icons

**res/values**
- App styles and themes
- Color details
- Localized strings (texts used in app UI)

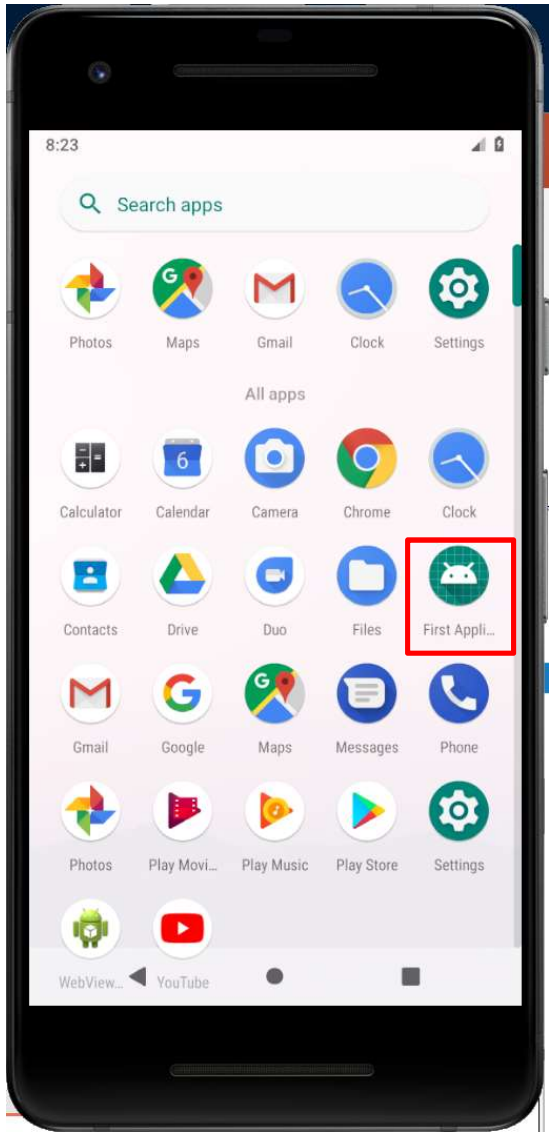
# App. icon



사각형
아이콘

원형
아이콘

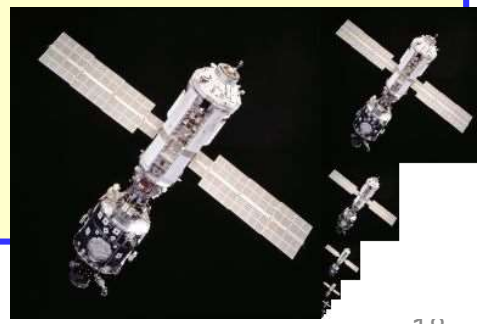


# Drawable 과 mipmap 차이 (1/2)

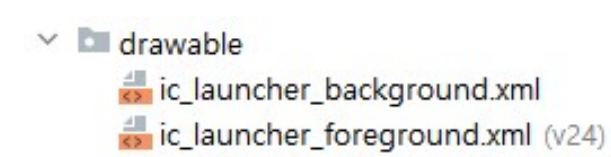
- **drawable**
  - For bitmap files (PNG, JPEG, or GIF), 9-Patch image files, and XML files that describe Drawable shapes or Drawable objects that contain multiple states (*normal*, *pressed*, or *focused*).

- **mipmap**
  - For **app launcher icons**
  - Android 4.2(API 17)부터 도입
  - The Android system retains the resources in this folder (and *density-specific folders* such as **mipmap-xxxhdpi**) regardless of the screen resolution of the device where your app is installed.
  - This behavior allows launcher apps *to pick the best resolution icon* for your app to display on the home screen.

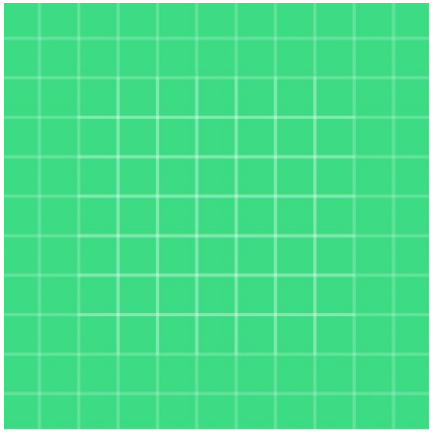
  - **mip-map**이란 무슨 뜻인가요?
    - 원본 이미지 축소판의 집합



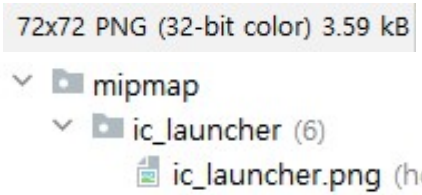
# Drawable 과 mipmap 차이 (2/2)

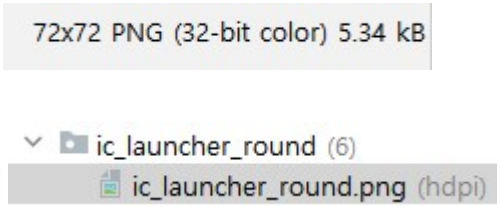



Drawable : xml 파일
수정 가능한 이미지 파일



mipmap : png 파일(이미지)
완성된 이미지(화면의 픽셀 밀도에 따라 선택됨)

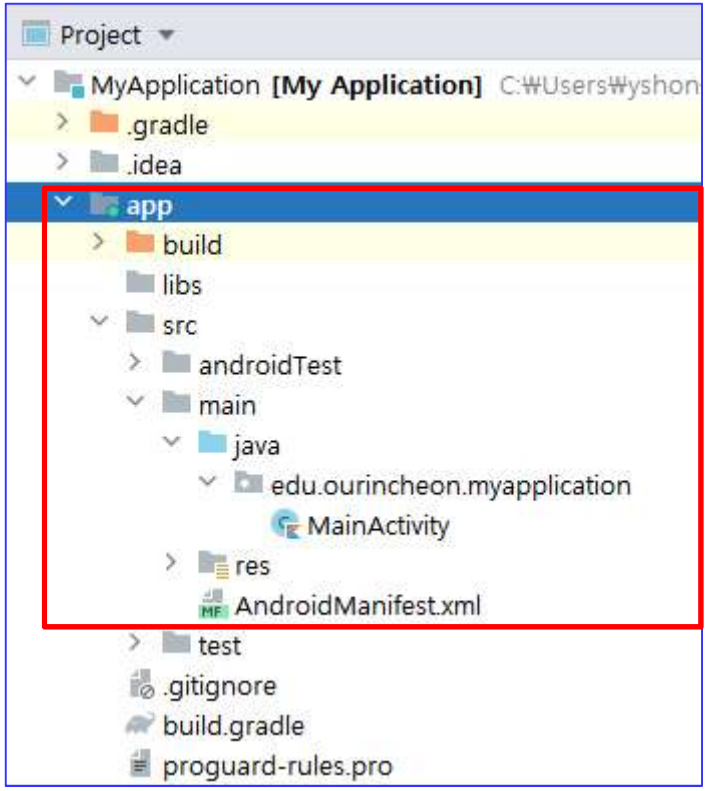


# What to do next?

- **Android Studio의 Project Window**
  - Drawable
  - App. Icon
    - Drawable, mipmap
  - **Project와 module**
  - **AndroidManifest.xml**

# project와 module

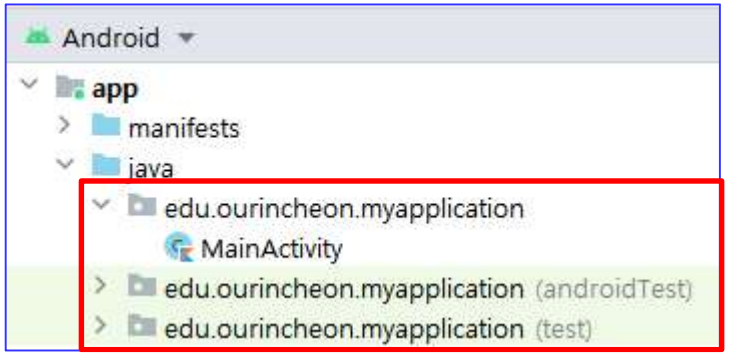


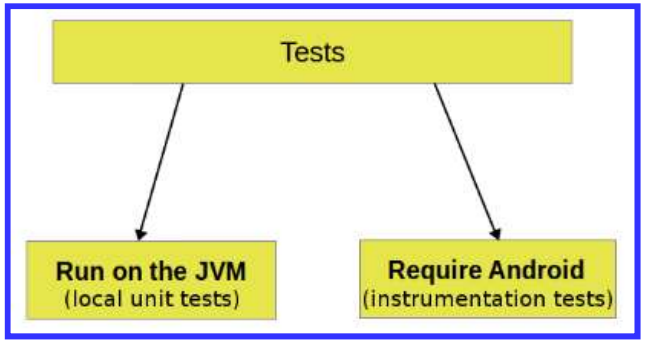

File > New > **New Module** ...
을 선택해서 새 모듈을
만들어 보자!

- project와 module은 **Android Studio** 용어
- **프로젝트 (project)**
  - project는 여러 개의 module로 이루어짐.
- **모듈 (module)**
  - 모듈 = 앱(App.)
  - module은 프로젝트에 포함

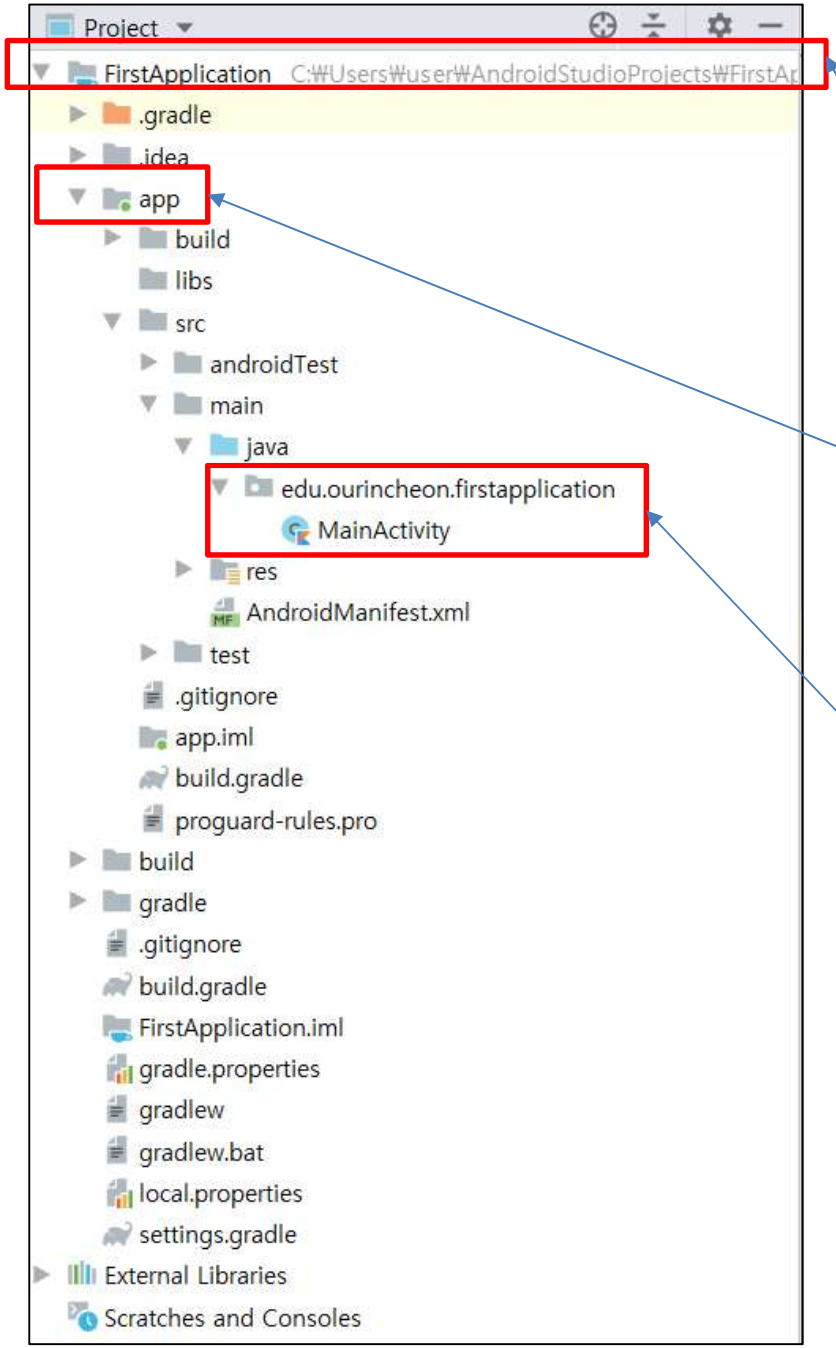
# 왜 이렇게 폴더가 많이 생길까요?

- **app>java 폴더에 생긴 하위 폴더들**
  - (package) **androidTest**
    - for unit tests that involves android instrumentation.
    - **To test code that *use Android framework***
  - (package) **test**
    - for pure unit test that do not involve android framework.
    - **To test code that are *pure java classes***



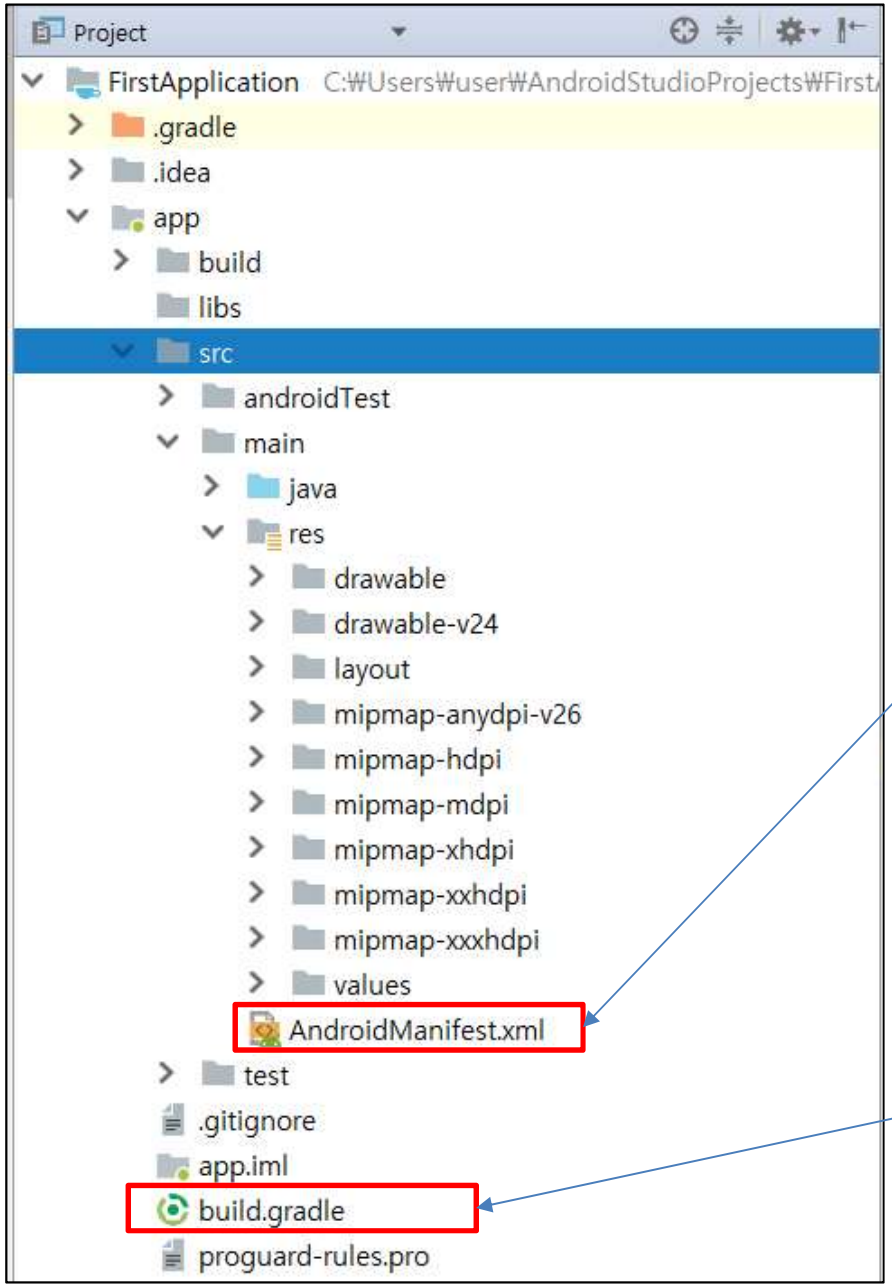

참고 사이트 http://www.vogella.com/tutorials/AndroidTesting/article.html
https://developer.android.com/studio/test/index.html




**FirstApplication**
- Parent Project Name
- Contains sub-projects and its files

**app**
- Sub-project
- Also known as a module

**edu.ourincheon.firstapplication**
- Package name
- Contains Java or Kotlin files



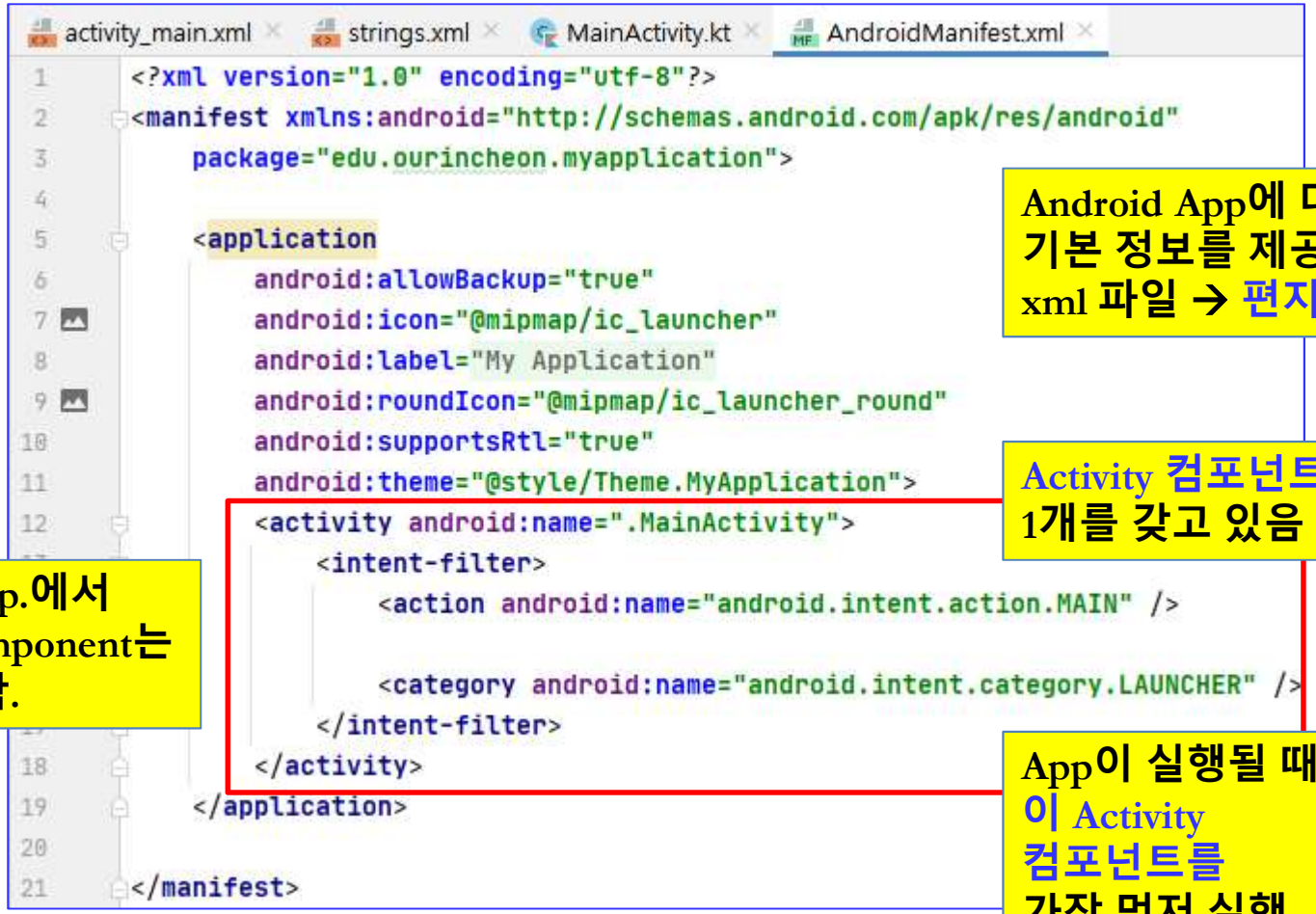

**AndroidManifest.xml**

- contains application components details
- Declaration of Activity, Service, BroadcastReceiver and ContentProvider
- Define necessary permission
  - USES INTERNET, USES CAMERA
  - READ SD CARD, etc.
- It is like summary of the application

**build.gradle**

- Build configuration
- Plugins to be used
- External libraries or dependencies to be included

# AndroidManifest.xml



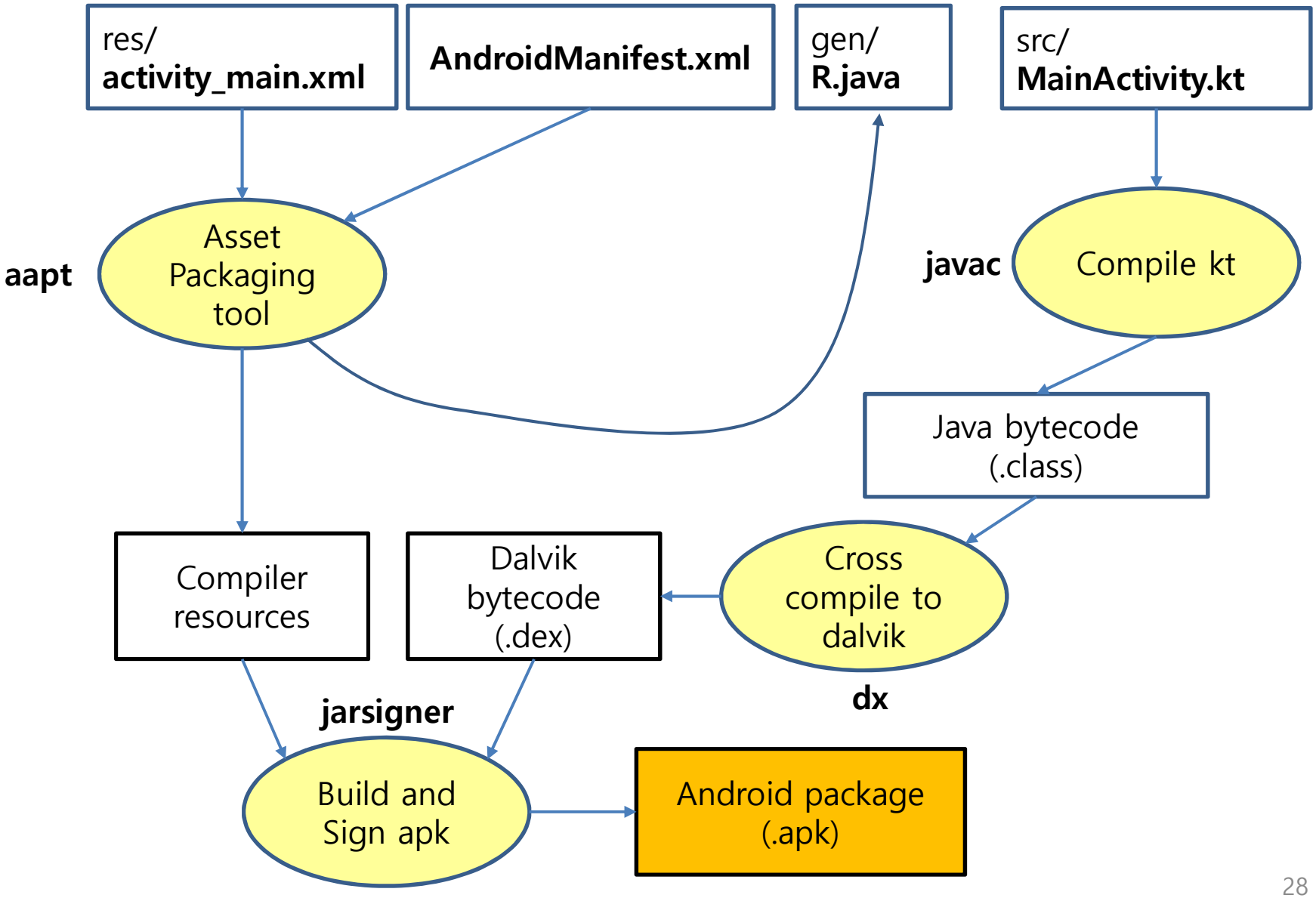
# What to do next?

- Android app.은 어디서부터 실행될까?
- Android project를 apk 파일로 build하는 과정
- **Model View Controller (MVC)**

# Android App.은 어디서부터 실행될까?

- **Android App.은 main() 메소드가 없다.**
  - 컴포넌트 중 UI를 갖는 Activity부터 먼저 실행
    - activity가 여러 개 있으면
      - 가장 먼저 실행할 activity를 Manifest 파일에서 지정
  - Activity 클래스에서는
    - **onCreate**( ) 메소드를 가장 먼저 실행.

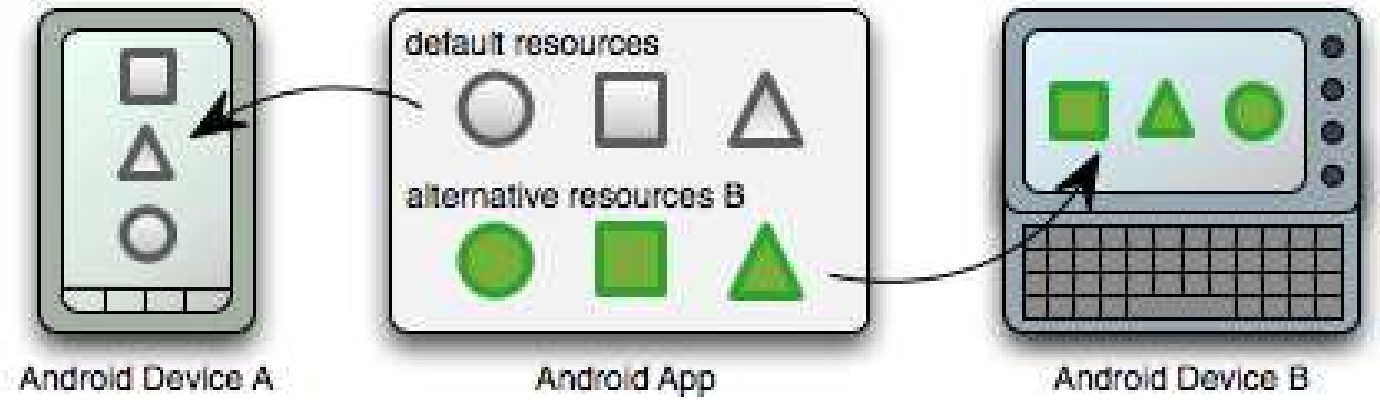
# Android project Build 과정



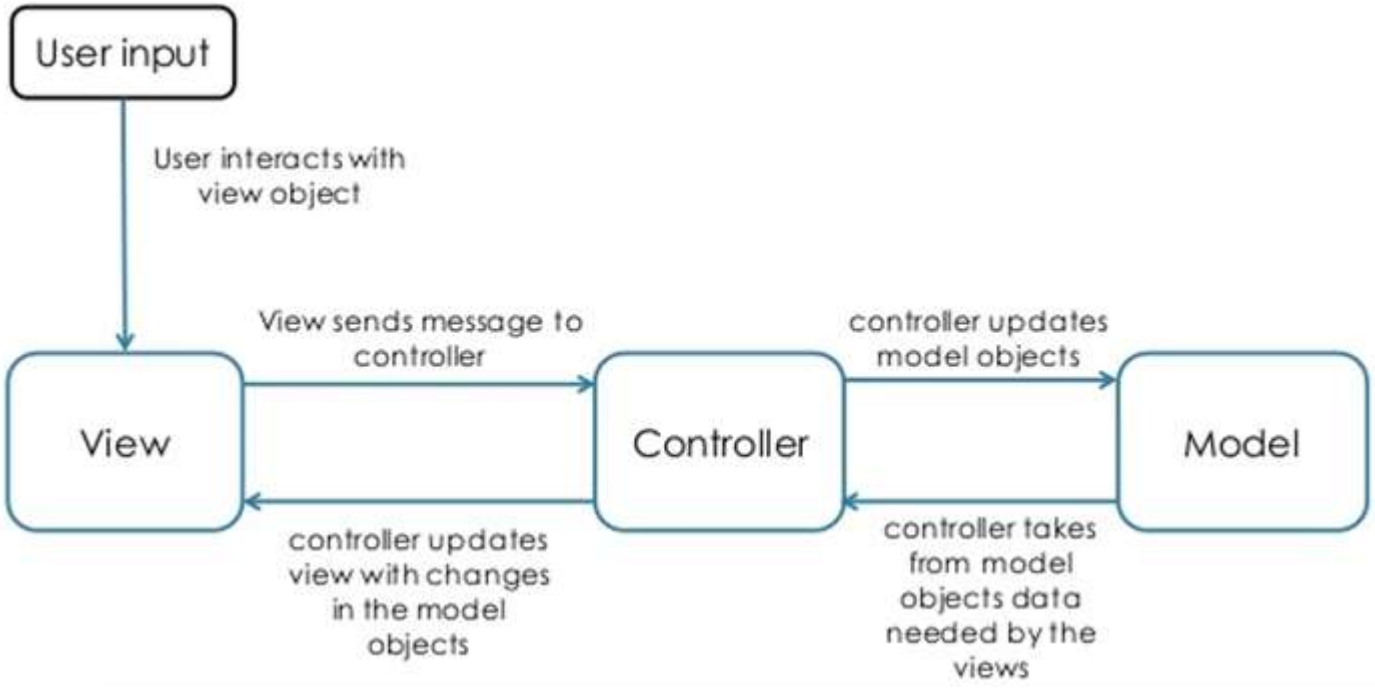
# 코드와 리소스를 분리

- Android가 다양한 장치에 탑재되면서
  - 언어나 화면 크기에 따라 리소스를 다르게 하는 것이 필요

- Android에서는 XML을 이용하여 UI를 설계하는 방법을 선호
  - App.의 UI와 business logic을 분리



# Model-View-Controller (MVC)




**activity_main.xml**

**MainActivity.kt**

**데이터 또는 Business logic**