



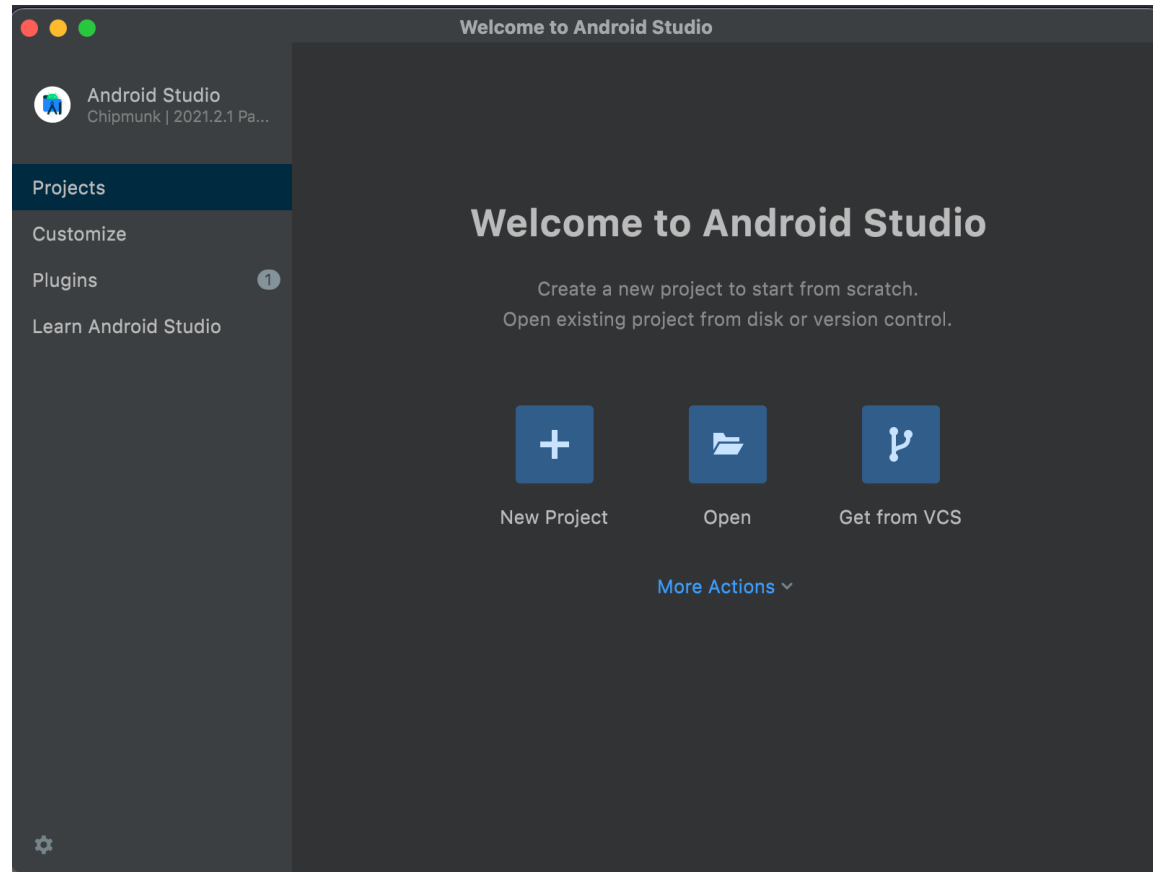
Kotlin

for Android App Development

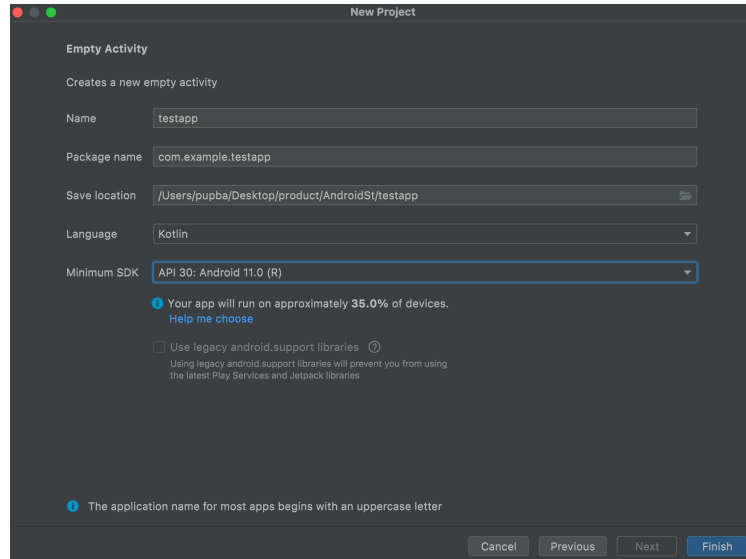
Make by Jung-Kwang-Won

안드로이드 스튜디오 설치

- ❑ <https://developer.android.com/studio/install?hl=ko> 에서 자신의 OS에 맞는 안드로이드 스튜디오 설치하기



프로젝트 생성

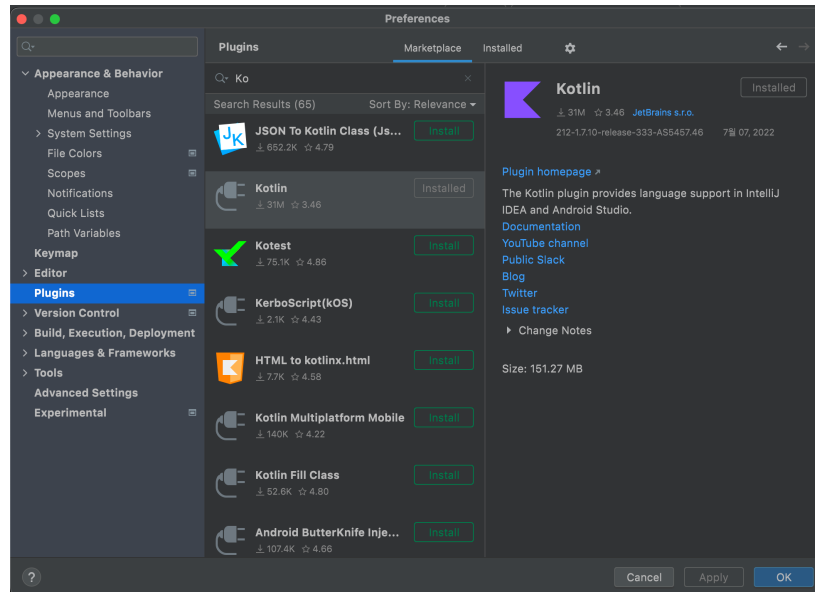


□ 설치가 완료되면 프로젝트를 만들어야 함.

- new project -> 원하는 템플릿 선택 -> 프로젝트 이름, 저장위치, 개발 언어, 안드로이드 버전을 선택 -> 완료

□ Kotlin Plugin 설치

- Setting -> Plugins -> Kotlin 검색 -> Kotlin 설치 -> OK 클릭

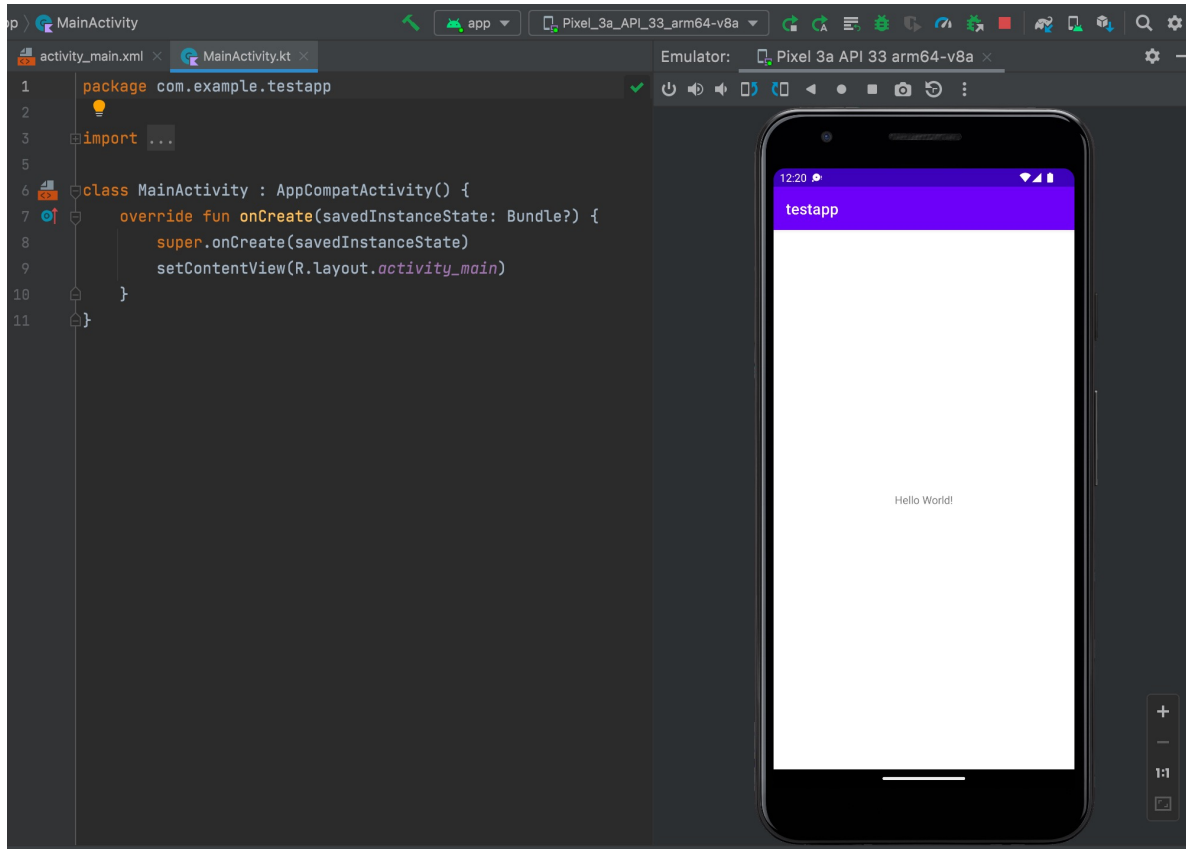


테스트해보기

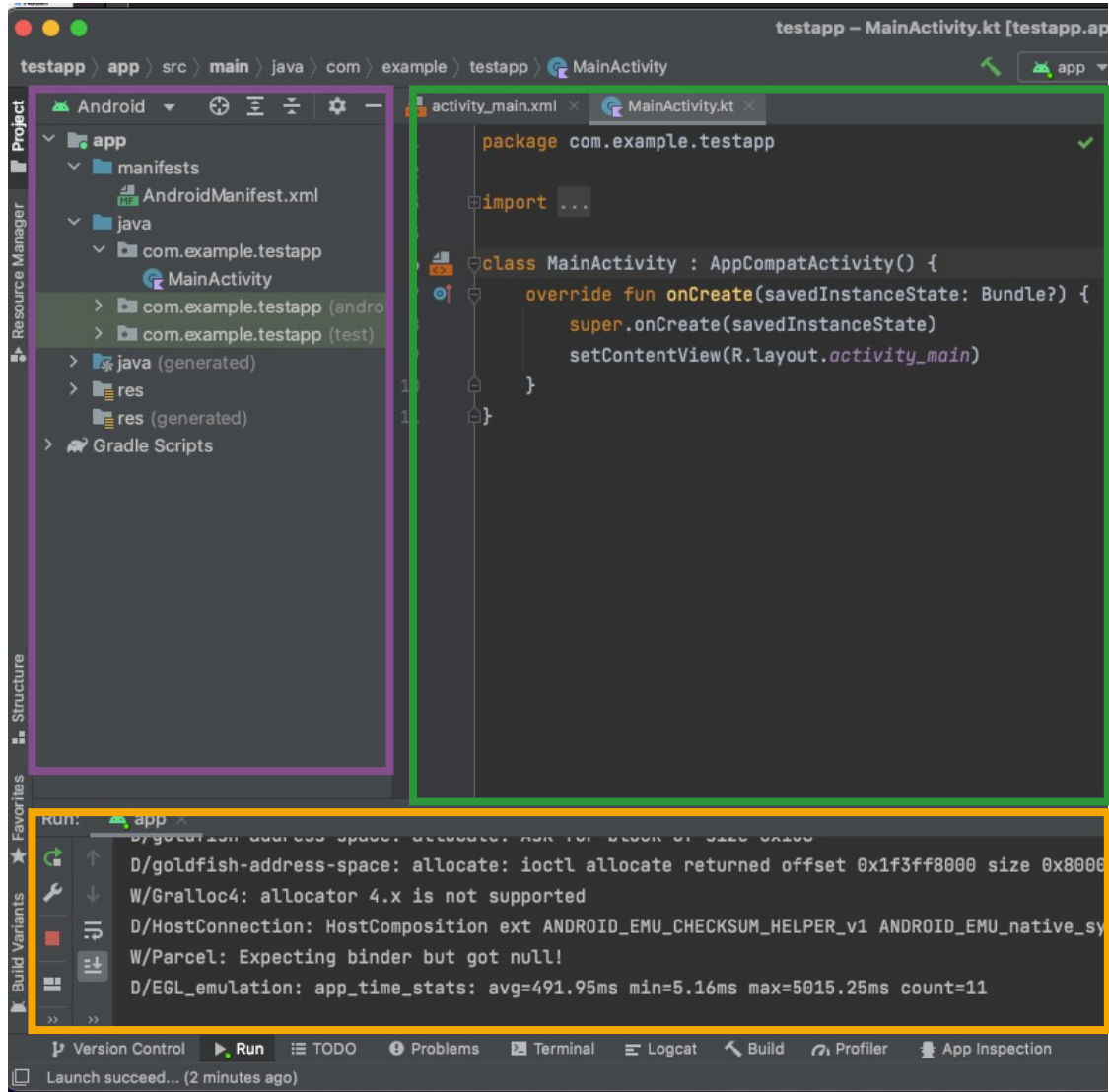
□ 오른쪽 위에 녹색 “▶” 버튼을 클릭하여 Emulator 실행.

□ 내 안드로이드폰에서 실행

- Device Manager -> Physical -> Pair using WiFi 누르기 -> 내 폰과 같은 WiFi에 접속 -> pair using QR code 창 띄워 놓기
- 핸드폰에서 개발자 모드 -> 무선 디버깅 키기 -> 무선 디버깅 누르기 -> QR 코드로 기기 페어링 눌러서 QR 코드 스캔 -> ▶ 눌러서 내 기기에서 확인.



UI 설명



□ Project tab

- project 하위 디렉토리 표시
 - ✓ Android의 GUI, Back-end, Manifests , Asset등을 관리함.
- Gradle Scripts 디렉토리 표시
 - ✓ 안드로이드 프로젝트의 빌드 정보, 라이브러리 등을 관리함.

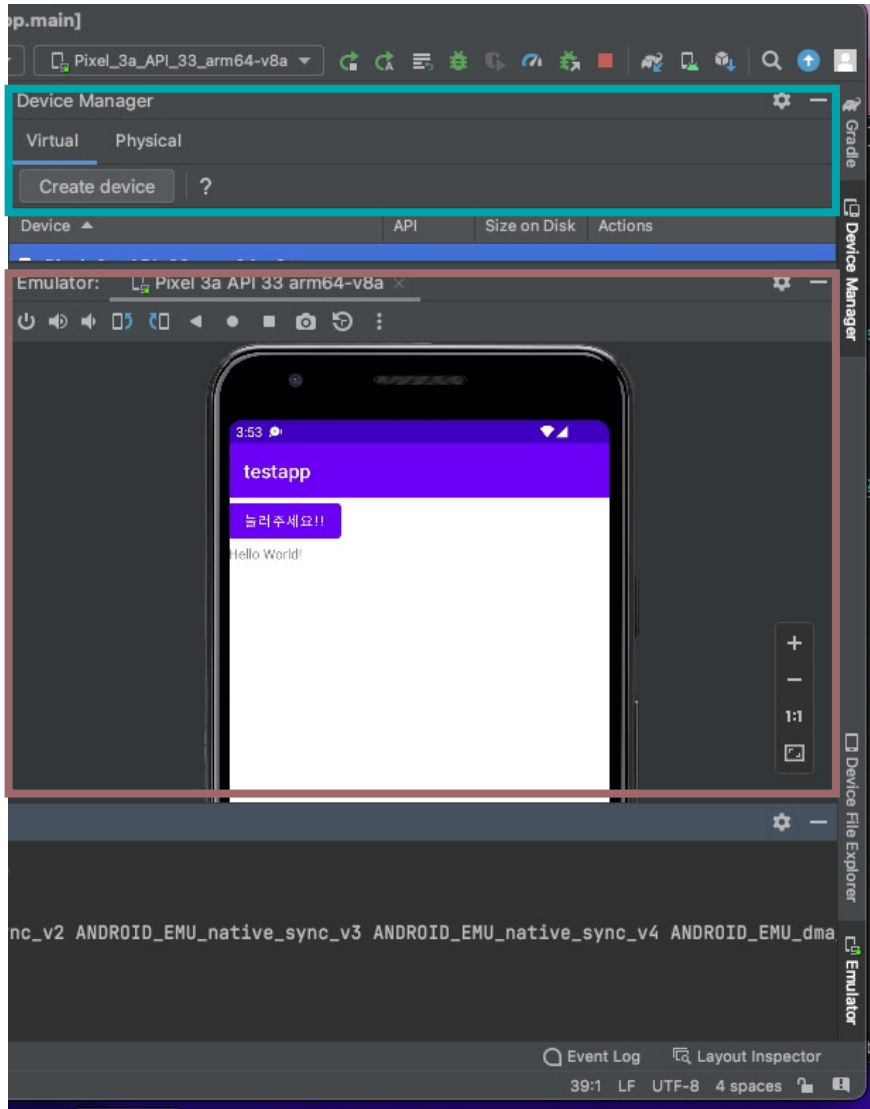
□ Work tab

- 파일을 선택하면 작업 영역이 열림.

□ Terminal tab

- Android Studio의 실시간 log 또는 에러 내용을 보여줌.

UI 설명



□ Device Manager tab

- Emulator를 선택하거나 설치 또는 외부 기기와 연결할 때 사용하는 탭.

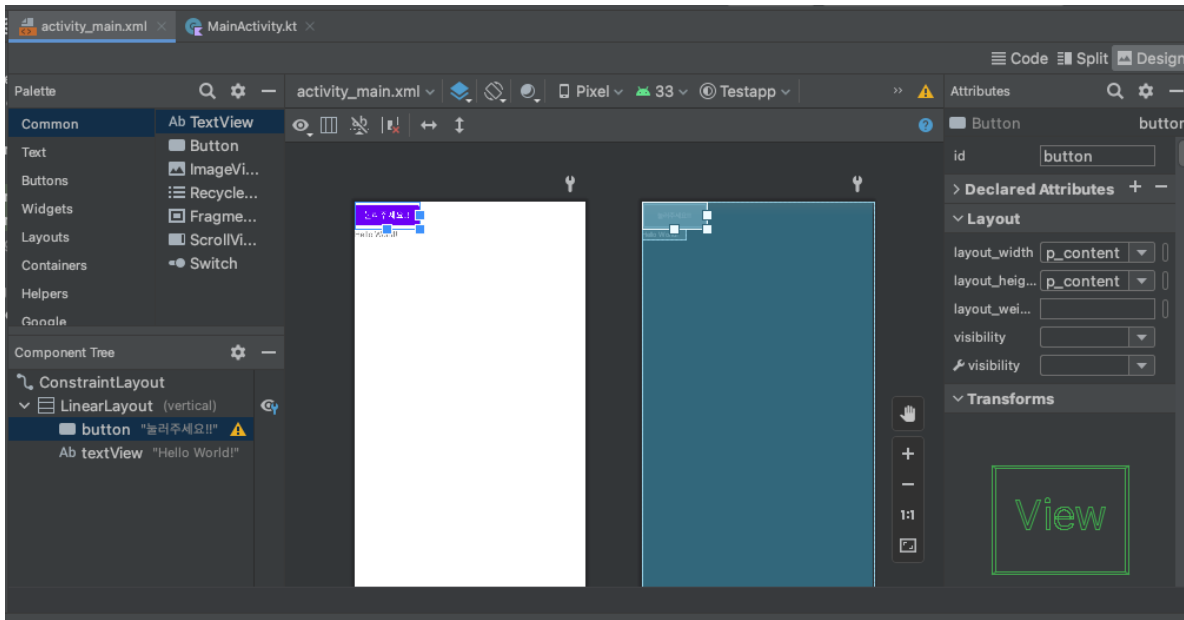
□ Emulator tab

- 빌드된 App을 테스트하는 탭.

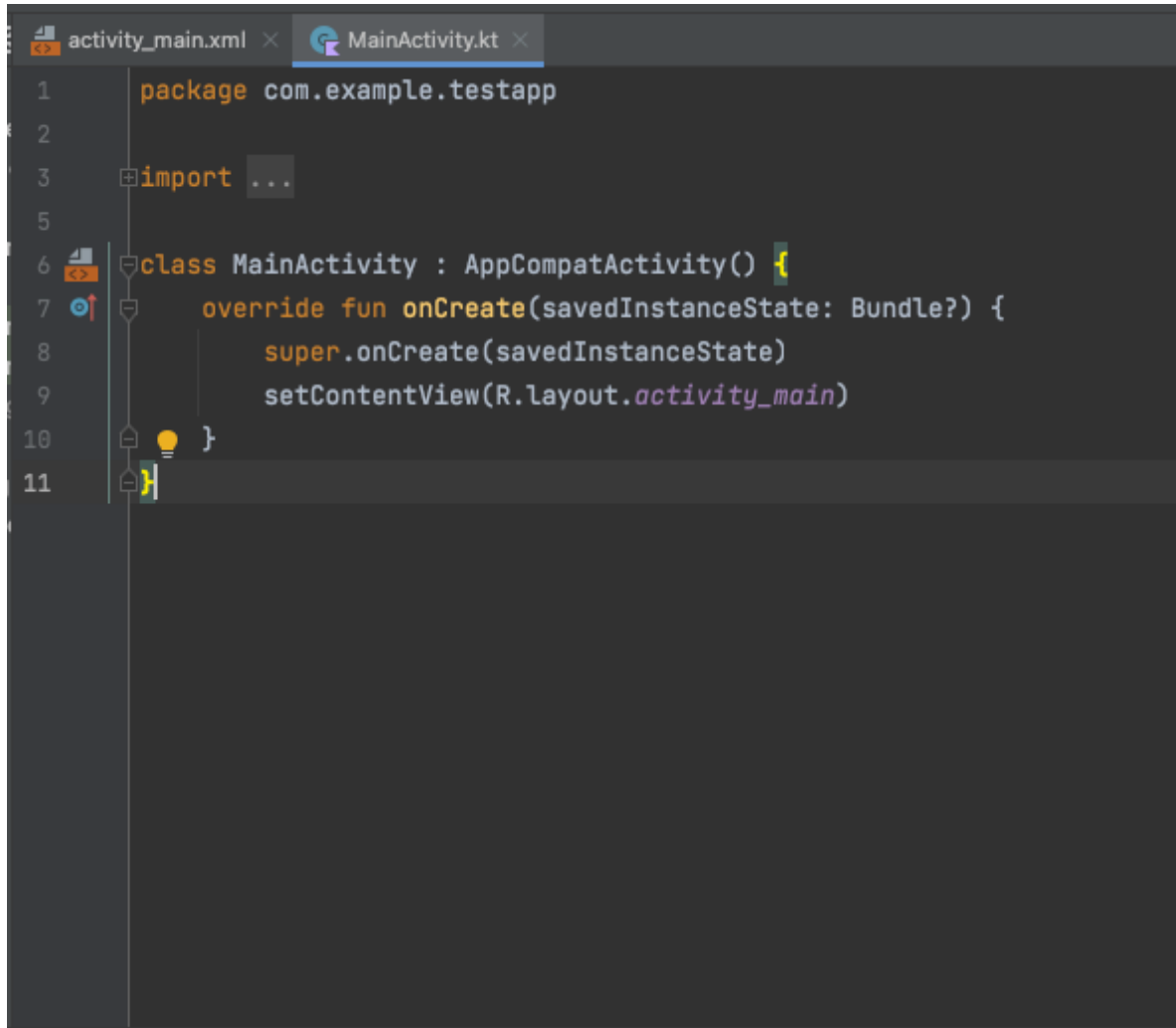
파일 설명

□ xml 파일

- GUI를 담당하는 파일.
- Drag-And-Drop 또는 Hard Coding으로 GUI를 배치, 생성, 조작을 함.
- "activity_main.xml"이 메인 뷰
- 뷰마다 한개 씩 존재함.
- .kt 또는 .java 파일과 연결됨.
- Palette에서 뷰를 가져옴.
- Attributes에서 View의 속성을 조작.



파일 설명



```
1 package com.example.testapp
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10    }
11 }
```

□ .kt(or .java) 파일

- Back-end 담당(App의 눈에 보이지 않는 구역에서 동작하는 기능을 담당.)
- View의 Event Trigger 작성.
- import로 package를 가져와 사용 가능.

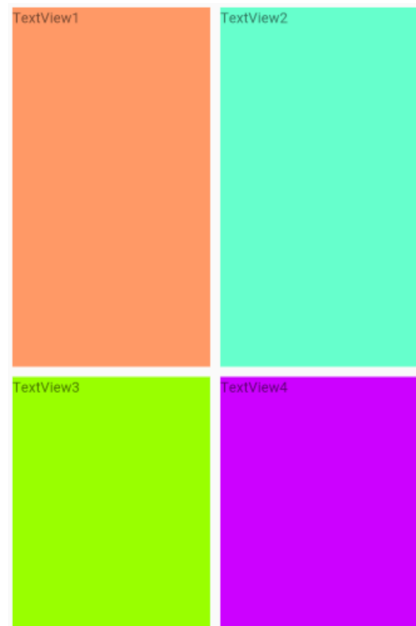
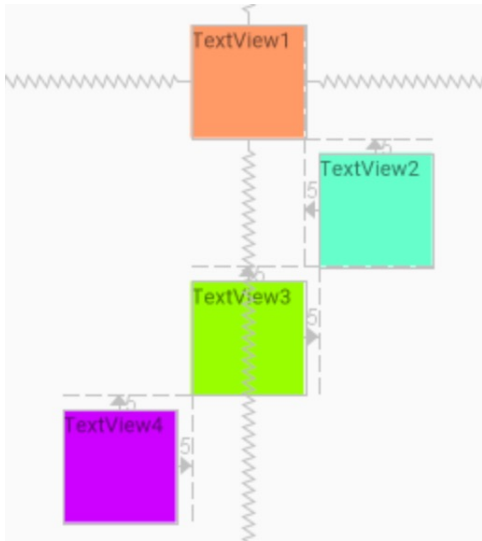
뷰의 기본 속성 정리

- ❑ **id** : 뷰의 아이디, 이벤트 처리시 사용.
- ❑ **text** : 보여질 텍스트.
- ❑ **textAlignment** : 텍스트의 위치 지정.
- ❑ **textColor** : 텍스트의 색.
- ❑ **textSize** : 텍스트의 크기(sp단위, Scale Independent Pixels)
- ❑ **textStyle** : 텍스트의 스타일(normal, bold, italic)
- ❑ **fontFamily** : 폰트를 설정함.

뷰의 기본 속성 정리

- ❑ `layout_width` : 뷰의 가로 크기.
- ❑ `layout_height` : 뷰의 세로 크기.
- ❑ `layout_weight` : 뷰의 비율을 지정함.
- ❑ `layout_margin` : 뷰의 마진을 설정함.
- ❑ `background` : 리소스 또는 색을 배경에 설정.
- ❑ `backgroundTint` : material-button 일 때 배경 설정.
- ❑ `foreground` : 리소스 또는 색을 뷰 자체에 설정.
- ❑ `clickable` : 버튼 눌리 듯이 뷰 설정.

레이아웃

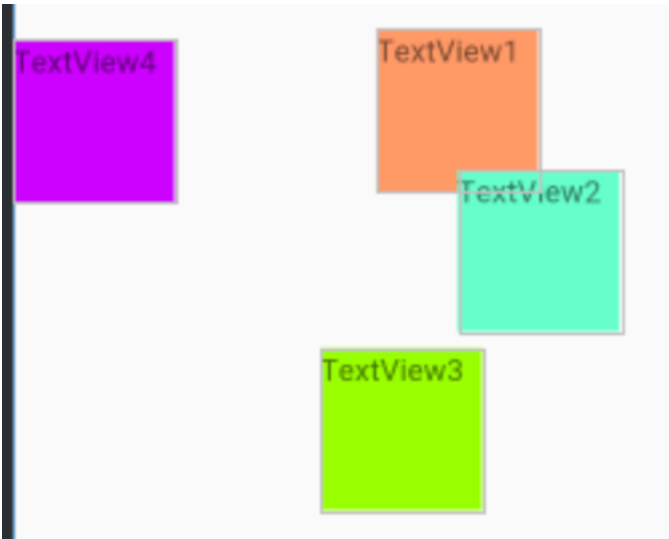
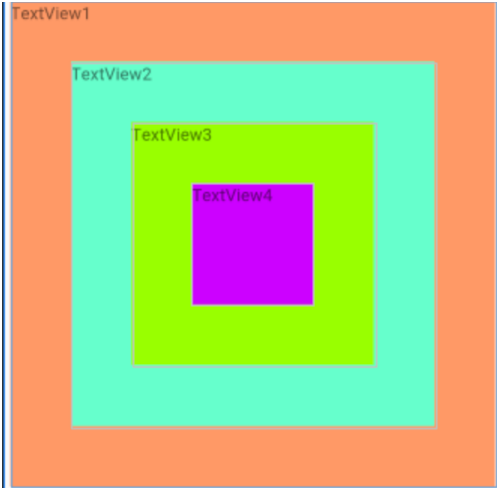


□ 레이아웃의 종류

- **Linear Layout** : 가로나 세로로 일렬로 나열할 때 사용.
- **Relative Layout** : 뷰를 특정 뷰나 부모의 위치에 맞게 배치할 수 있음.
- **Table Layout** : 뷰들을 표 처럼 배치할 수 있음, 표의 크기와 모양, 테두리 색 변경이 가능함.

□ 레이아웃 안에는 뷰를 넣을 수 있음.

레이아웃

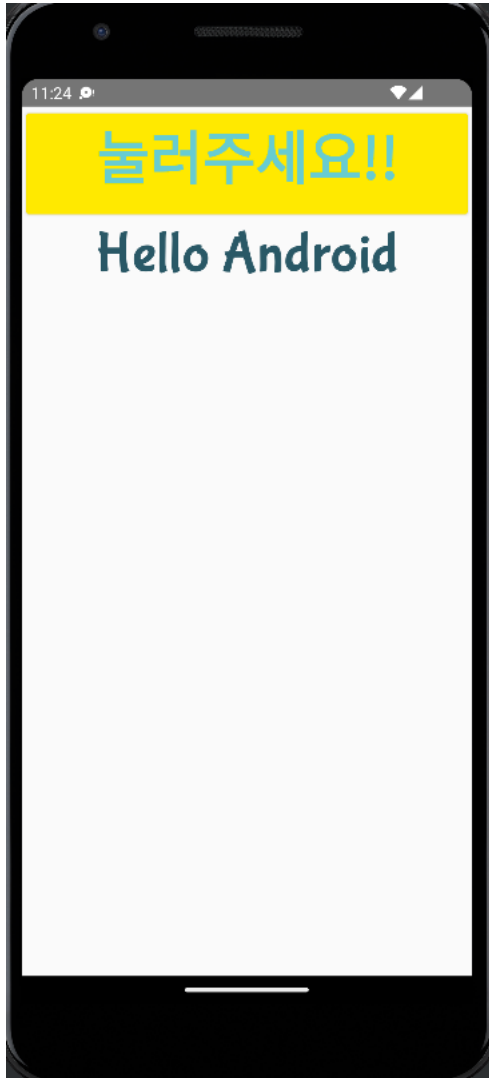


□ 레이아웃의 종류

- **Frame Layout** : 뷰들을 액자 처럼 배치할 수 있음(겹침)
- **Drawer Layout** : 뷰들을 서랍에 꺼내듯 배치할 수 있음, 메뉴나 프로필 정보를 담는 곳으로 많이 씀.
- **Constraint Layout** : 뷰의 위치를 반드시 강제적으로 지정해주어야함. 세세하게 배치할 수 있음.

□ 레이아웃안에 또 레이아웃을 넣을 수 있음.

title bar, action bar 없애기



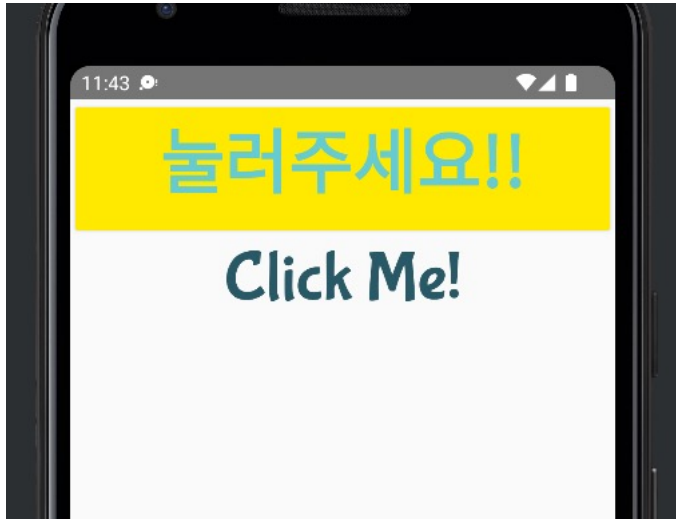
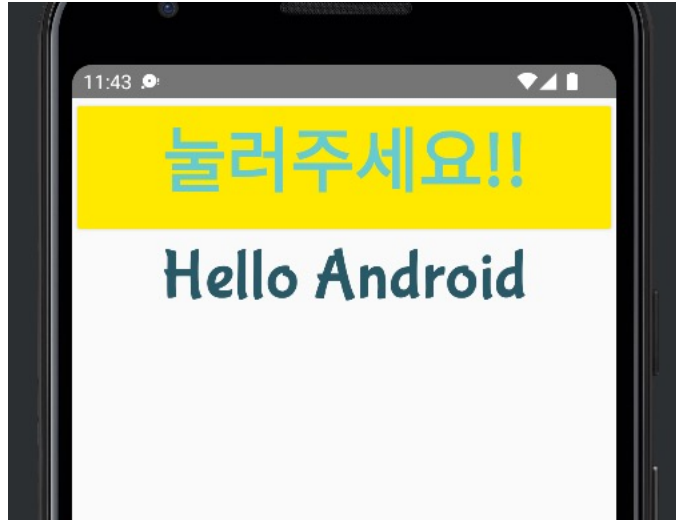
□ res -> value -> themes.xml(또는 styles.xml) 열기 후 다음과 같이 수정.

```
<!-- parent=Theme.MaterialComponents.~를 밑 처럼 변경 -->
<style name="Theme.Testapp" parent="Theme.MaterialComponents.Light.NoActionBar">
  <!-- 밑에 두 줄 추가 -->
  <item name="windowActionBar">false</item>
  <item name="windowNoTitle">true</item>
```

□ AndroidManifest.xml에서
〈application〉 태그 안에
"android:theme"를 다음과 같이 변경.

```
android:theme="@style/Theme.AppCompat.Light.NoActionBar"
```

버튼 이벤트



□ 먼저 버튼의 id를 얻어 와야함.

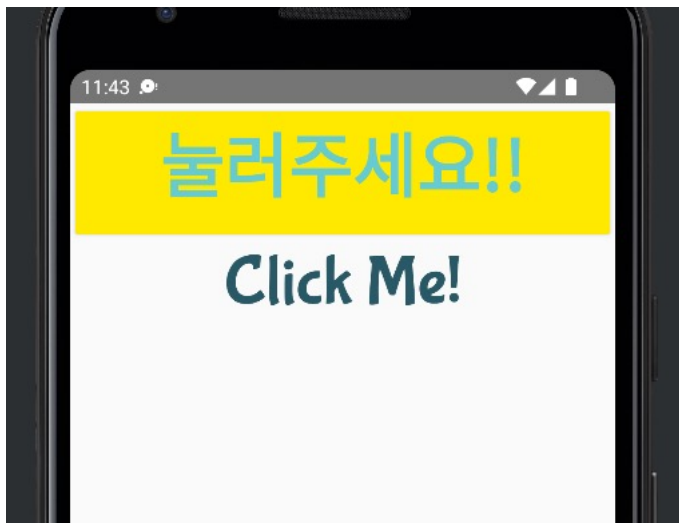
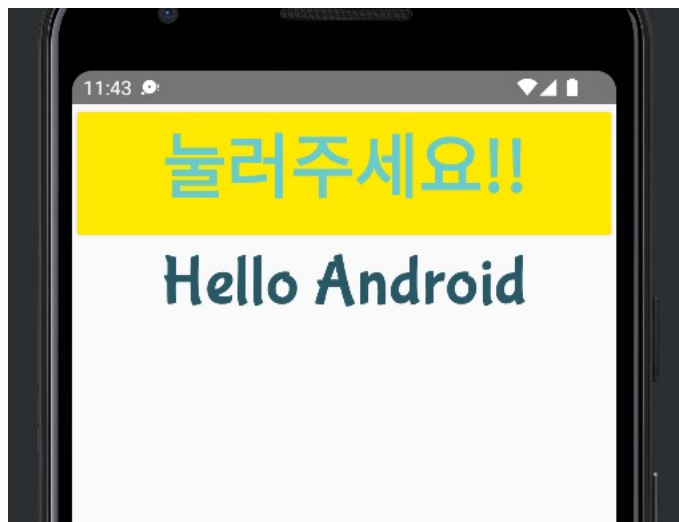
- id를 얻어와서 객체로 상수에 저장

```
import android.widget.Button
// onCreate 메서드 안에 있어야함.
val btn1 = findViewById<Button>(R.id.btn_1)
```

- 상수를 사용하여 버튼 이벤트를 작성하여 버튼을 제어함.
- ✓ ex) 버튼을 눌렀을 때의 이벤트 메서드

```
btn1.setOnClickListener{
    // 버튼을 눌렀을 때 실행할 내용
}
```

텍스트 이벤트



□ 먼저 버튼의 id를 얻어 와야함.

- id를 얻어와서 객체로 상수에 저장

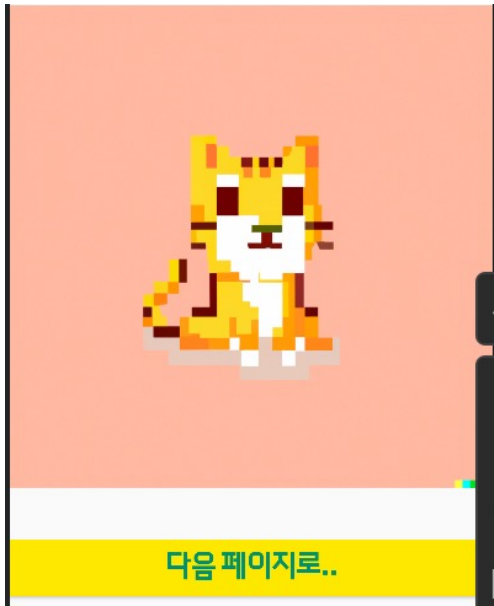
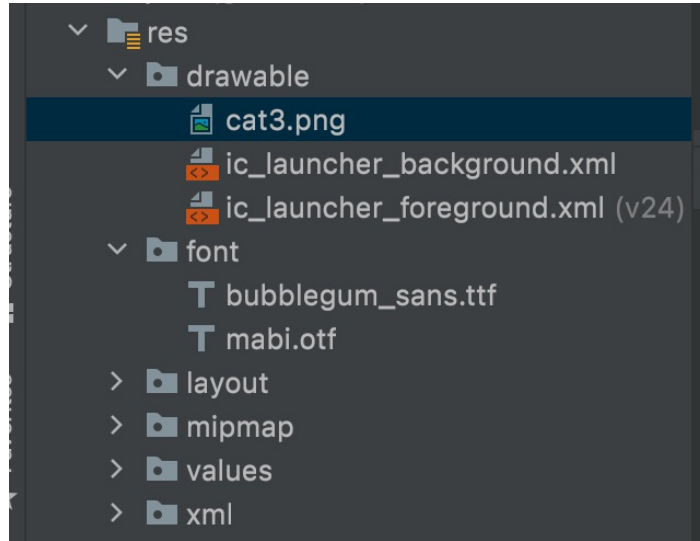
```
import android.widget.TextView
val text = findViewById<TextView>(R.id.change_text)
```

- 객체를 사용하여 텍스트를 제어함.

✓ ex) 텍스트를 변경

```
// 기본 text 값 == "Hello Android"
// 코틀린은 setText 대신 text = "" 로 해줘야
text.text = "Click Me!" // text 변경
```

Resource 추가

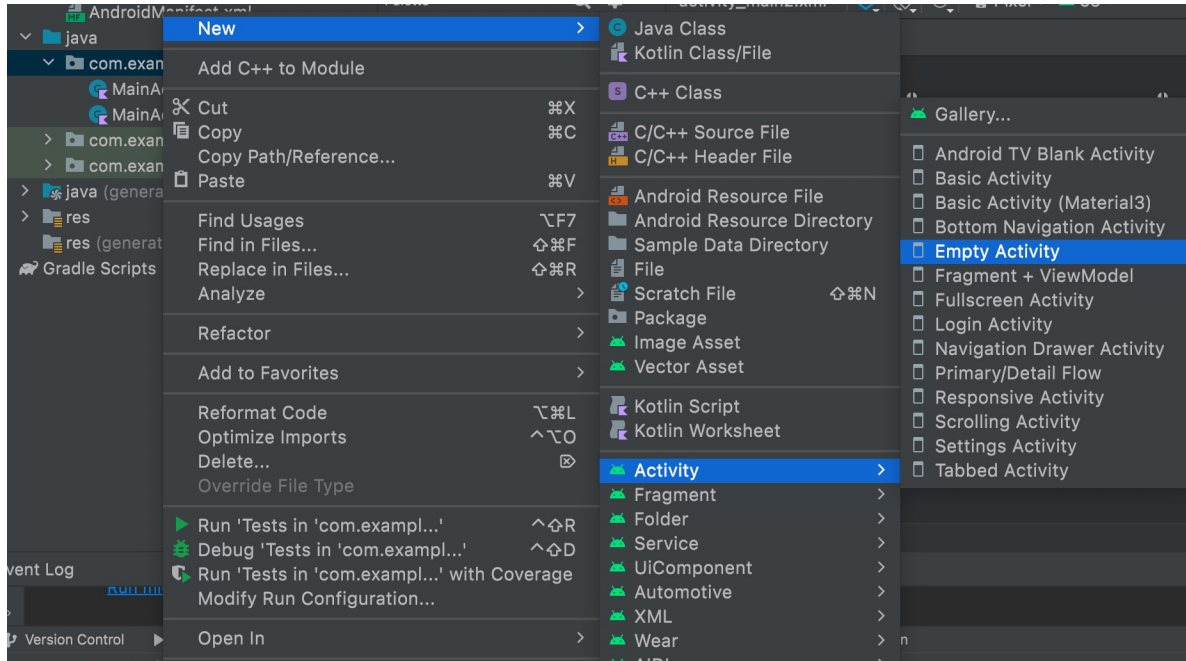


□ Resource

- 이미지, 폰트, 아이콘 등
- res 디렉토리 하위에 저장됨.
- 원하는 Resource를 res 아래에 드레그 앤 드랍하거나 new에서 원하는 형태의 리소스를 넣으면됨.

□ Resource는 View 사용시 사용가능함.

여러개의 Activity



□ Activity를 추가할 수 있음.

○ Activity == 스크린

□ 생성

○ package에서 마우스 우클릭 -> New -> Activity -> 원하는 템플릿 선택 -> Activity 이름 입력 후 finish

○ xml, .kt 파일 생성됨.

✓ MainActivity 처럼 사용하면 됨.

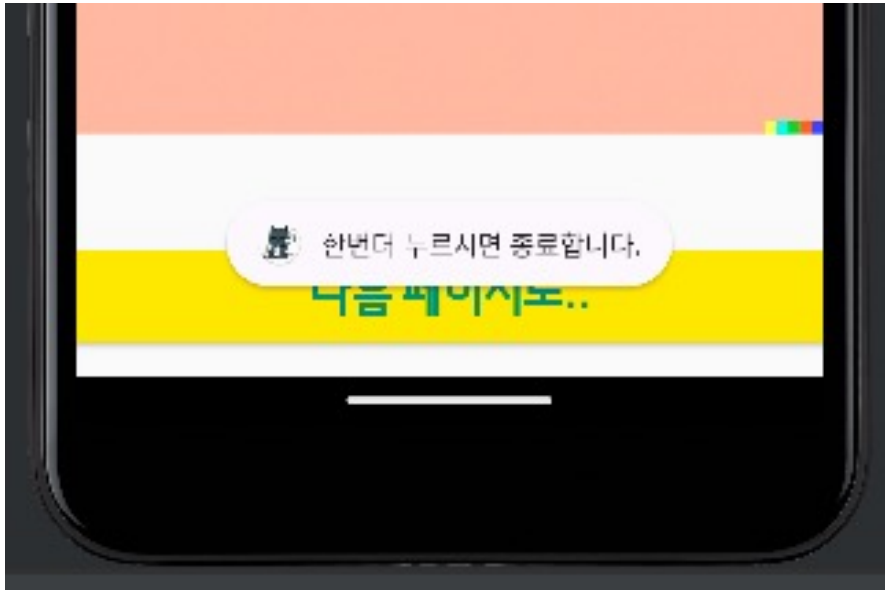
□ Intent 클래스로 화면 전환.

○ Intent 클래스로 객체를 만든 다음.

○ startActivity(Intent 객체)로 화면 전환.

```
버튼 객체.setOnClickListener{  
    val intent = Intent(this, 이동할 Activity 이름::class.java)  
    startActivity(intent)  
}
```

Tip - 한번 더 누르면 종료



□ 멤버 변수 2개 선언.



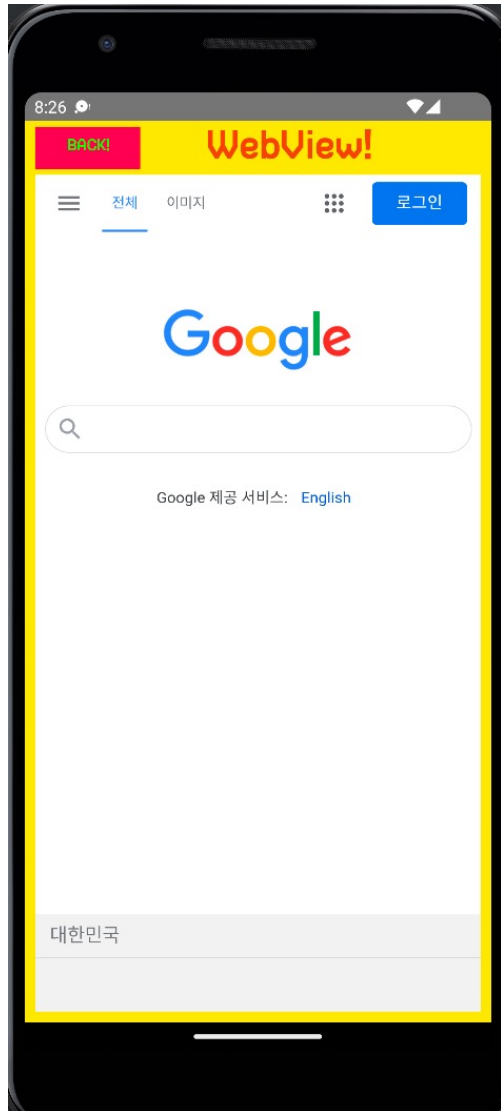
```
private val finishtimeed: Long = 1000  
private var presstime: Long = 0
```

□ onBackPressed() 메서드를
override함.



```
override fun onBackPressed() {  
    var tempTime: Long = System.currentTimeMillis()  
    var intervalTime: Long = tempTime - this.presstime  
  
    if(intervalTime in 0..finishtimeed){  
        moveTaskToBack(true) // 태스크를 백그라운드로 이동  
        finishAndRemoveTask() // 액티비티 종료 + 태스크 리스트에서 지우기  
        System.exit(0)  
    }  
    else{  
        this.presstime = tempTime  
        Toast.makeText(applicationContext, "한번 더 누르시면 종료합니다.", Toast.LENGTH_SHORT).show()  
    }  
}
```

Tip - WebView



❑ WebView란?

- 브라우저 처럼 사용할 수 있는 뷰.

❑ AndroidManifest.xml 수정

```
//<manifest ~> 밑의 코드 추가, 인터넷 권한 부여 🙏🙏🙏  
<uses-permission android:name="android.permission.INTERNET" />  
// ~  
//</manifest>
```

❑ .kt 코드 작성

```
import android.webkit.WebChromeClient  
import android.webkit.WebView  
import android.webkit.WebViewClient  
val webView = findViewById<WebView>(R.id.webView)  
webView.settings.apply {  
    javaScriptEnabled = true  
    domStorageEnabled = true  
    setSupportMultipleWindows(true)  
}  
webView.apply {  
    webViewClient = WebViewClient()  
    webChromeClient = WebChromeClient()  
    loadUrl("https://www.google.com")  
}
```

APP 이름 변경

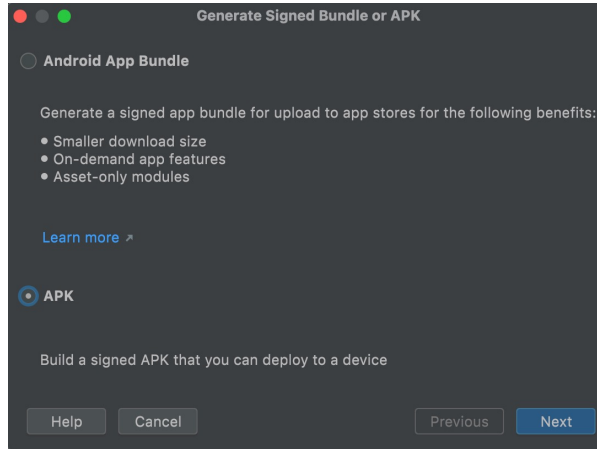
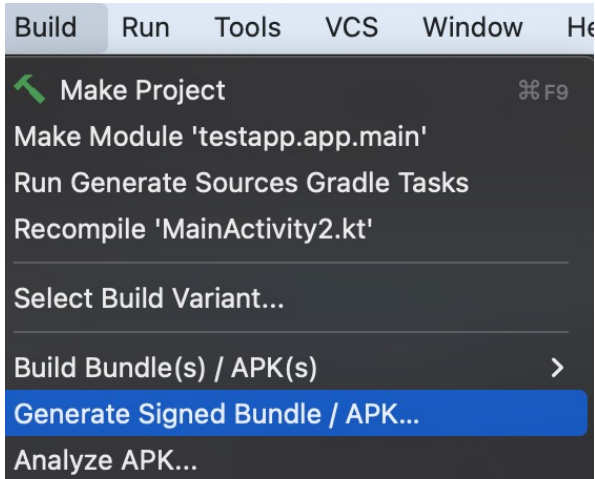
```
AndroidManifest.xml
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data
    android:fullBackupContent="@xml/backup
    android:icon="@mipmap/ic_launcher"
    android:label="Cats"
```

```
strings.xml
Edit translations for all locales in the translations editor.
1 <resources>
2   <string name="app_name">Cats</string>
3   <string name="back">Back!</string>
4   <string name="webview">WebView!</string>
5   <string name="todo">TODO</string>
6   <string name="todo2">TODO</string>
7 </resources>
```

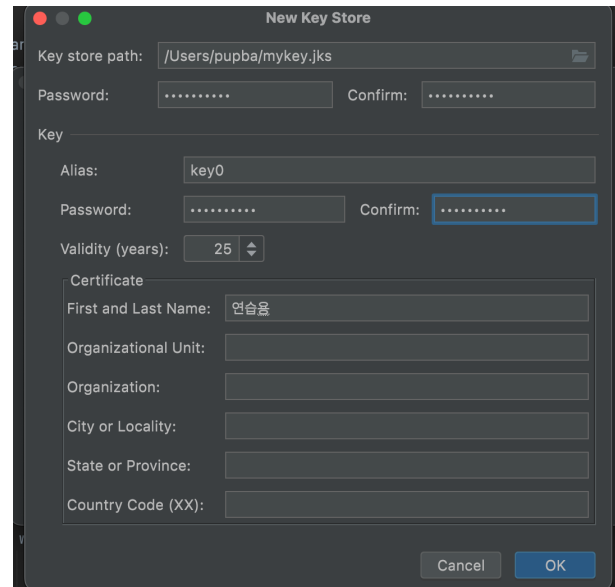
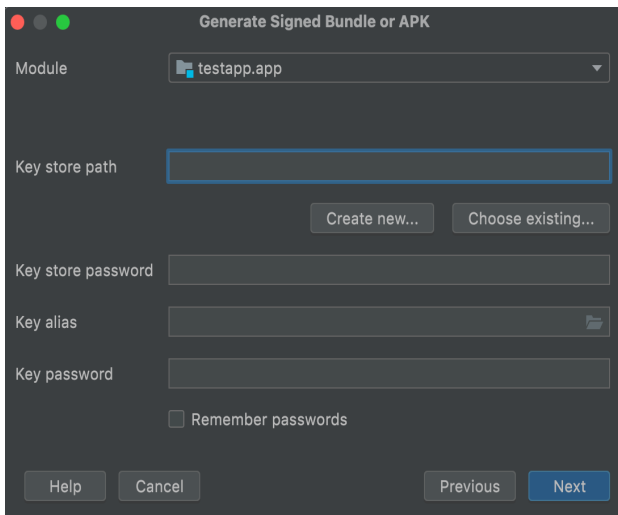
□ 처음 프로젝트를 만들 때의 이름이 기본적으로 App의 이름이 되지만 이것을 변경할 수 있음.

- AndroidManifest.xml에서 `android:label = ""`을 변경하여 이름을 바꾸는 방법.
- `res -> values -> strings.xml`에서 `<string name = "app_name">`의 바로 옆 문자열을 변경

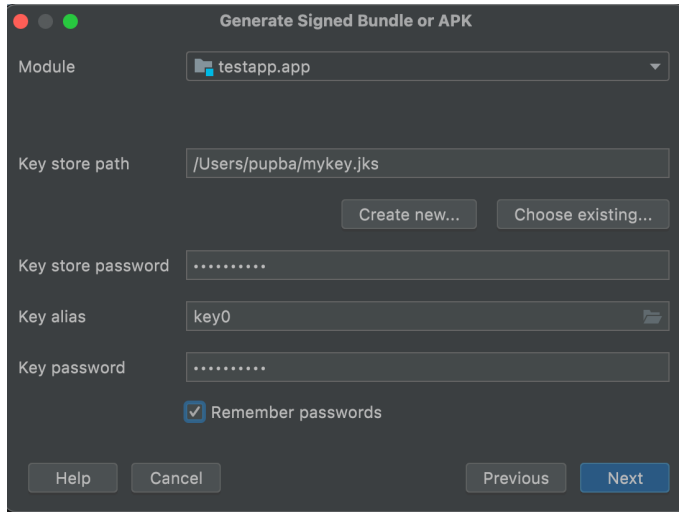
APK 파일 빌드



- ❑ Build -> Generate Signed Bundle/APK -> Build APK(s)
- ❑ APK 선택 -> Next
- ❑ Key store 선택 하고 password 입력
 - 없는 경우 --> Key Stone 만들기
 - Key store path, password, alias, certificate 등 작성 후 OK 클릭으로 생성
 - path와 password는 꼭 기억해야함.

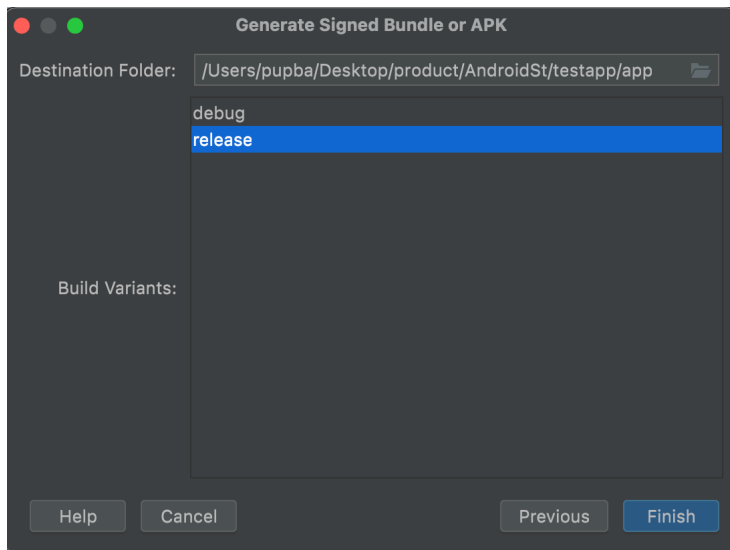


APK 파일 빌드

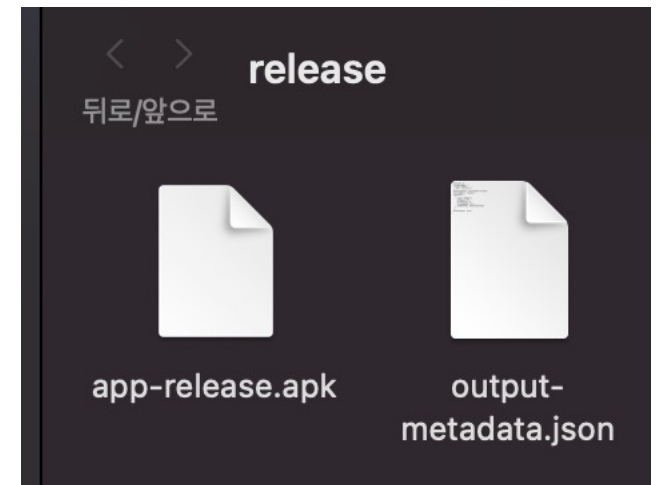


□ 입력 후 Next 누르기 -> release 선택 후 Finish

□ 완료하면 location 눌러서 디렉토리 열고 APK 파일 배포하기.



Generate Signed APK
APK(s) generated successfully for module 'testapp.app' with 1 build variant:
Build variant 'release': [locate](#) or [analyze](#) the APK.



Android Studio Views

□ View에 대한 공식 문서

- <https://developer.android.com/reference/kotlin/android/view/View>

□ 사용법을 모르는 View의 사용법은 Google에 검색해서 찾아보자.

- ex1) 안드로이드 스튜디오 WebView 코틀린
- ex2) 안드로이드 스튜디오 WebView 자바
- ex3) Android Studio WebView Kotlin
- ex4) Android Studio WebView Java

□ 설치된 SDK의 버전과 스마트폰의 버전에 따라 일부 기능이 동작을 안할 수도 있으니 잘 알아보고 개발 할것.