WaffleStudio 2023
Rookies 21.5 Seminar

Android Seminar 0
2023.09.08 (Fri)

Instructor : 양주현 (@JuTaK97) / TA : 송동엽 (@eastshine2741)

세미나장 소개

- 양주현 (@JuTaK97)
- 자연대 지구환경과학부 16학번 (컴공 복수전공), 23년 2월 졸업
- 와플스튜디오 19.5기 루키, SNUTT 안드로이드 개발 (2022.02~)
- VCNC(타다) 에서 23년 4월부터 8월까지 안드로이드 개발자로 근무
- 9월부터 토스페이먼츠에서 안드로이드 개발자로 근무 예정

- 아이폰 사용경험 없음 ~~(하지만 맥북은 반필수)~~

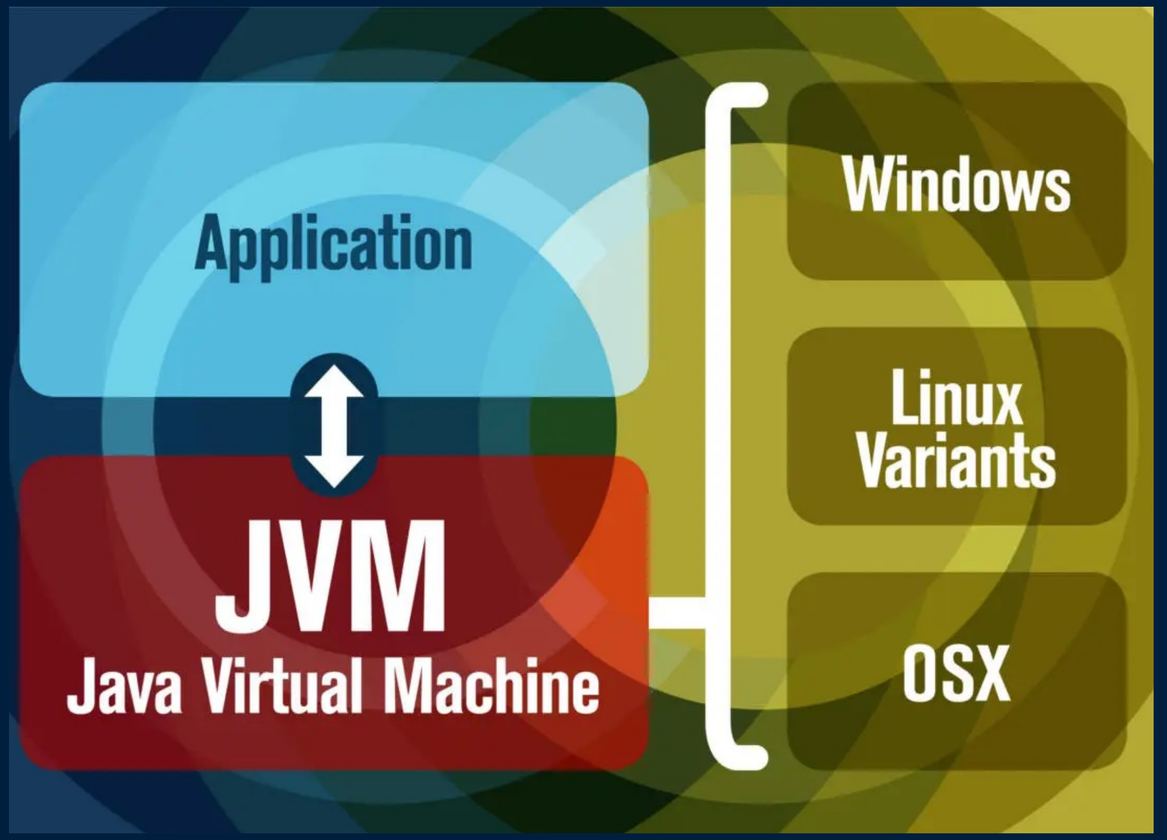
Kotlin 기초 (1)

코틀린은 Java와 함께 JVM 위에서 동작하는 언어입니다.

- JVM이란?

  자바 프로그램을 실행하는 가상 머신

- Java와 Kotlin은 상호 운용성이 존재해서 Java ~ Bytecode ~ Kotlin 전환이 거의 100% 가능하며, 문법 또한 유사하다.



Kotlin 기초 (2)

- 널 안정성

  Nullable 타입

  Safe call

  Elvis Operator

  스마트 캐스팅

```
fun main() {
    val a: Int? = calculate1()
    val aa = a
    if (a != null) {
        getInt(a)
        getInt(aa)
    }
}
```

Kotlin 기초 (3)

- 타입 추론

  타입을 명시적으로 선언하지 않아도 컴파일러가 타입을 분석하여 유추해 준다!

```
sealed class Animal {
    data class Dog(val name: String): Animal()
    data class Cat(val name: String): Animal()
    data class Bird(val name: String): Animal()
}

fun play(animal: Animal) {
    when (animal) {
        is Animal.Dog -> {

        }
        is Animal.Cat -> {

        }
    }
}
```

Kotlin 기초 (4)

- 함수형 프로그래밍 (람다)

함수를 일급 시민으로 취급한다.

일급 시민이란?

함수를 변수에 저장하거나, 파라미터 혹은 리턴 값으로 사용할 수 있고 동적으로 생성할 수 있다.

```
val adder: (Int, Int) -> Int = { a, b -> a+b }
val multiplier: (Int, Int) -> Int = { a, b -> a*b }

val composition: ((Int, Int) -> String, (String) -> Int) ->
((Int, Int) -> Int) = { f1, f2 ->
    val f1f2: ((Int, Int) -> Int) = { a, b -> f2(f1(a, b)) }
    f1f2 // 람다의 마지막 statement는 반환값
}
```

Kotlin 기초 (5)

```
class Button(val touchEvent: () -> String) {
    private fun speak(word: String) {
        println(word)
    }

    fun click() {
        val result = touchEvent()
        speak(result)
    }
}
```

```
fun downloadFromNetwork() : String {
    return ""
}

fun whatTimeIsItNow(): Long {
    return System.currentTimeMillis()
}
```

```
fun main() {
    // 동적으로 함수를 생성할 수 있다!
    val touchEvent: () -> String = {
        downloadFromNetwork() + whatTimeIsItNow()
    }

    // 만든 "함수"를 인자로 전달했다.
    val myButton = Button(touchEvent = touchEvent)

    // 어느 시점에 click 함수를 부르면??
    myButton.click()
}
```

안드로이드 개발에 대해… (1)

기존에 사용하던 언어들은?

- 대부분 아는(?) 단어로 구성되어 있고,

- 어떤 절차로 실행되는지 비교적 잘 이해하고 있습니다.

```c
#include <stdio.h>

int main() {
    printf("Hello world!");
}
```

```
$ gcc main.c
$ ./a.out
Hello world!
```

안드로이드 개발에 대해… (2)

앞으로 배울 코드는 생소합니다.

- main 함수가 없습니다. 어디서 실행이 시작되는 거지?

- Activity란? onCreate는?

- R.layout.activity_main의 정체는?

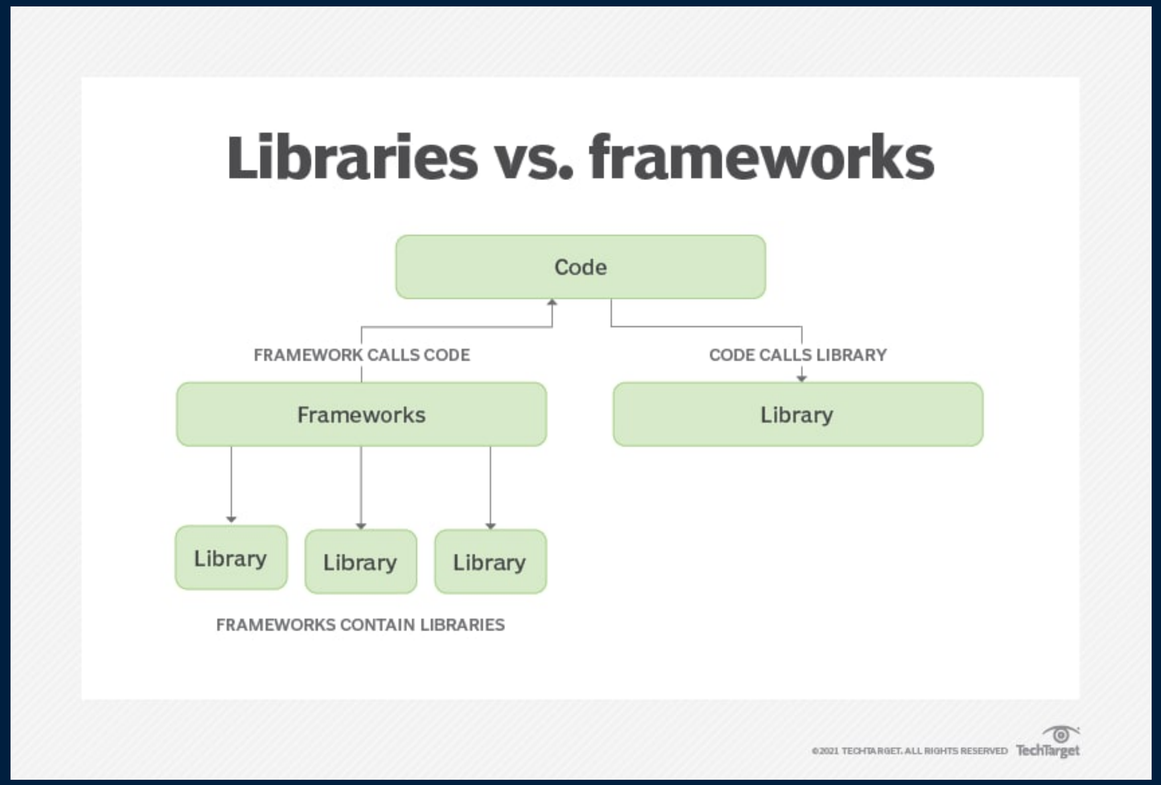
```
class MainActivity {
    override fun onCreate() {
        super.onCreate()
        setContentView(R.layout.activity_main)
    }
}
```

안드로이드 개발에 대해… (3)

우리는 Framework를 공부합니다.

Android Platform 이 이해할 수 있는 방식으로 내가 원하는 행동을 코드로 작성하면, 이 코드를 Platform 이 가져다가 실행시키게 됩니다.

즉, 우리가 작성하는 `MainActivity` 를 실행하는 주체는 우리가 작성하는 코드가 아닌 Android Framework 입니다.



안드로이드 개발에 대해… (4)

앞으로 우리는

- Android Platform 위에서 어떻게 하면 우리가 원하는 기능을 만들 수 있는지 배워보고

- 그 과정에서 일반적으로 좀 더 좋은 코드를 짜는 법과, 그런 방법이 어떻게 안드로이드 개발에 녹아있는지를 배우게 됩니다.

따라서 우리가 배우는 내용은 대부분 **원하는 기능을 구현하는 방법** 을 위주로 진행될 예정입니다.

1. 내가 원하는 UI 를 만드는 방법
2. 사용자 상호작용 (UX) 를 만드는 방법
3. 외부 서버와 통신하는 방법
4. 데이터를 저장하고 읽어오는 방법
5. 좀 더 **좋은** 코드를 작성하는 방법

안드로이드 기초 (1)

Android App Component

https://developer.android.com/guide/components/fundamentals?hl=ko

플랫폼과 상호작용할 수 있는 기본 구성요소

우리가 주로 사용할 구성요소는? Activity

- **Activity**
  - 사용자와 직접적인 상호작용이 이뤄지는 진입점 담당
- Service
  - 보통 화면에 보이는 상호작용 없이 뒷단의 작업 담당
  - e.g. 백그라운드에서 파일 다운로드
- Broadcast Receiver
  - 다른 App 이나 시스템과의 통신을 담당
  - e.g. 비행기 모드 전환 이라는 이벤트 수신
- Content Provider
  - App 내에서 존재하는 데이터 관리 담당
  - e.g. 사진첩 등

안드로이드 기초 (2)

Activity

우리는 Android Platform 위에 있습니다.

AppCompatActivity 클래스를 상속하는 MainActivity 클래스를 우리가 구현하면, Android Platform은 MainActivity 객체를 생성하고 onCreate 함수를 실행합니다.

우리는 클래스와 각 함수를 Android Platform 이 언제 어떻게 사용하는지 공부하고, 앱이 우리가 원하는 동작을 하도록 구현합니다.

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

안드로이드 기초 (3)

생명 주기 (Activity Lifecycle) 링크

onCreate과 같은 함수들은 Activity의 생명 주기와 관련된 함수입니다.

이 함수를 구현한다는 것은, Android Platform가 Activity를 생성하거나 제거할 때 우리의 앱이 무슨 일을 할 지 명시해 준다는 것입니다.

예를 들어 onCreate는 앱이 처음 시작될 때 한 번 불리는 함수입니다. 우리는 보통 이때 _화면에 보이는 UI 를 구성하고, UI 를 클릭할 때 발생하는 이벤트를 등록합니다.

안드로이드 기초 (4)

이제 이 코드를 어느 정도 이해했습니다.

모르는 부분이 어디가 남았나요?

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

안드로이드 기초 (5)

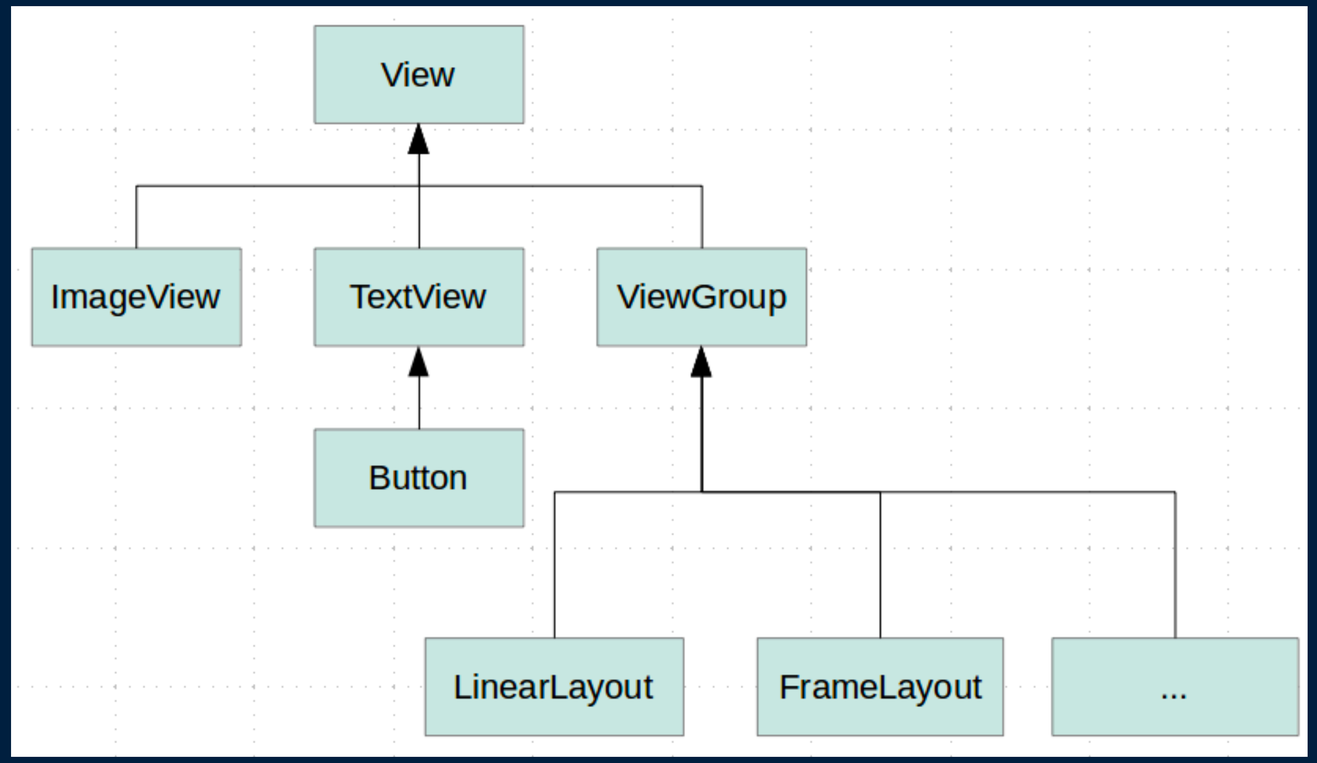
View

- 실제 Android 의 UI 를 구성하는 빌딩 블록

- 계층 구조로 이루어져 있다.

- 버튼, 텍스트 상자, 이미지, 레이아웃 등등

- View 는 스스로가 화면에서 어느 곳에 위치할 지, 자식들이 어디에 위치할지, 스스로 어떤 식으로 그려질 지 등을 결정한다.



안드로이드 기초 (6)

XML

- 데이터를 표현하는 하나의 언어 (HTML 친구)

- Android는 XML 파일을 설계도로 사용한다.

- xml 파일은 inflate 라는 과정을 통해 실제 접근 가능한 View 계층 구조 로 전환된다.

- 앞서 본 R.layout.activity_main 가 바로 이 xml 파일을 의미하고, setContentView 라는 함수를 이용해서 액티비티의 화면을 xml 파일 대로 구성한다.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, Android!" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me" />

</LinearLayout>
```

기초 UI (1)

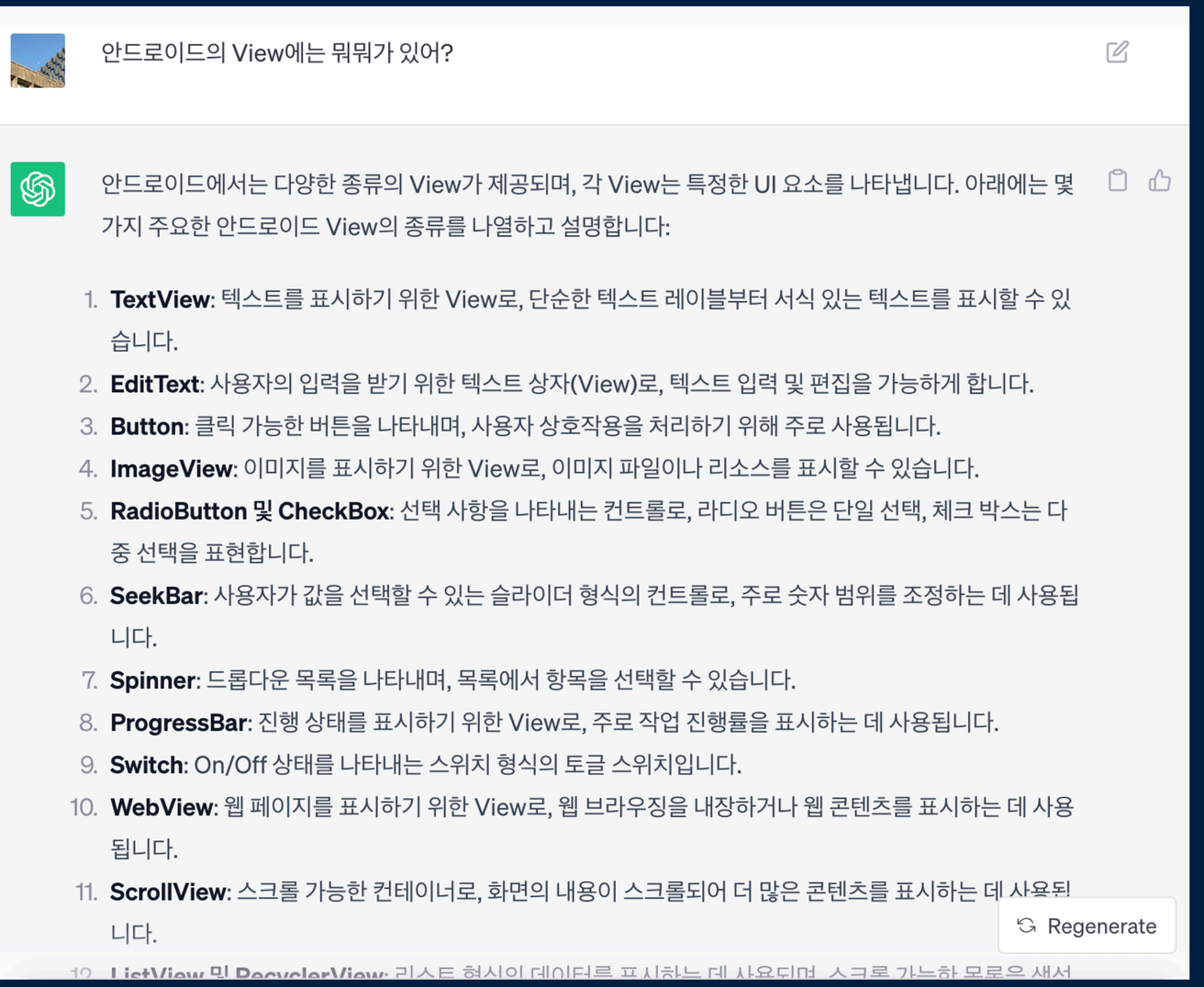
다양한 View를 알아보자!

주로 사용할 것 :

- **ConstraintLayout**, **LinearLayout**, **FrameLayout**

- **TextView**, **EditText**, ImageView, ScrollView, **RecyclerView**, WebView, ButtonView, …

직접 하나씩 써보세요! 쓰는법은 구글링! 과제에서 저 중에 몇몇 것들을 써야 할 지도…



기초 UI (2)

View의 다양한 attribute

- text, textColor, textStyle, …

- gravity, alpha, background, translationX, translationY, elevation, …

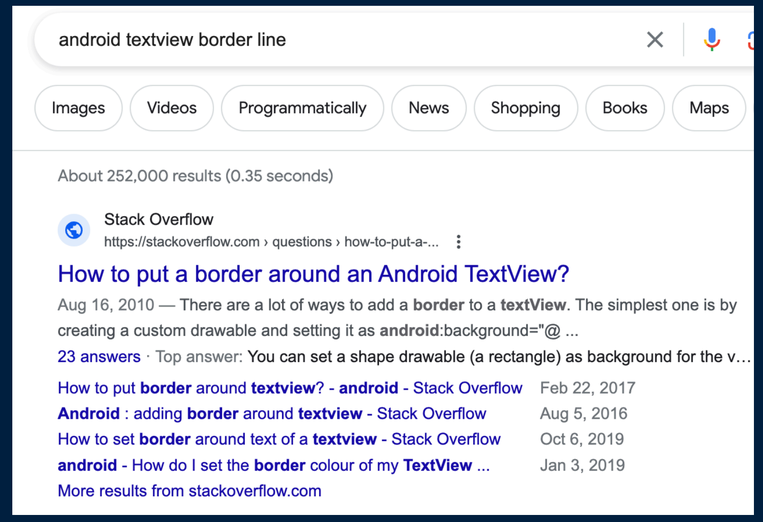
수많은 속성이 존재하고, 우리는 필요한 속성을 잘 지정하고 조합해서 원하는 형태의 View를 그리면 됩니다.

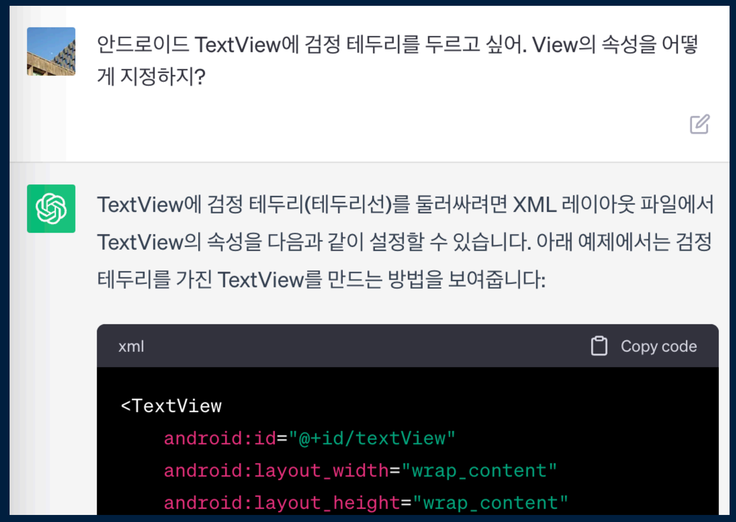
직접 여러 UI를 만들어 보고, 몸으로 겪어 보며 하나둘씩 모험을 통해 알아가시면 됩니다.

궁금한 게 있으면 우리의 친구에게 물어봅시다.

만약 우리의 친구의 대답이 시원찮더라도 여러분의 곁에는 세미나장과 조교가 있습니다.

질문하세요!





기초 UI (3)

ConstraintLayout

여러 View들 간 상대적인 위치를 지정합니다. 가장 자주 쓰고, 쓸모가 많고, 어려운 Layout

- layout_width, layout height

  match_parent, wrap_content, {숫자}dp, 0dp, …

- layout_constraint

  parent 또는 다른 View에 4방향으로 constraint를 걸어 준다. 팽팽한 갈고리를 걸어 놓는 느낌

직접 많이 해 봐야 느낌이 옵니다. 많은 경험을 통한 체득이 필요한 layout

기초 UI (4)

클릭 이벤트를 받아보자!

xml을 inflate하여 계층 구조로 만들어서 메모리에 올리는데, 이때 특정 View에 접근하려면 “이름”이 필요하다. 이것이 바로 **id!**

MainActivity의 onCreate 함수에서 findViewById 함수를 이용해 이 id를 갖는 View 객체를 가져올 수 있다.

터치가 감지되면 TextView 객체에 정의된 OnClickListener의 onClick 함수가 불린다. 우리가 할 일은 인터페이스의 내용(=람다)을 채워 주는 것 뿐이다.

```
<TextView
    android:id="@+id/text" <!-- 이 라인 추가 -->
    android:layout_width="wrap_content"
    android:layout_height=“wrap_content"
    <!-- 중략 -->
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)

var count = 0
val textView = findViewById<TextView>(R.id.text)
textView.setOnClickListener {
    count++
    textView.text = count.toString()
}
```

기초 UX (1)

화면 전환을 해 보자!

1. 또다른 액티비티를 만든다.

   SecondActivity와 activity_second.xml

2. AndroidManifest.xml 파일에 오른쪽 코드를 추가한다.

3. 클릭 리스너에 오른쪽 코드를 추가한다.

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

```
<activity
    android:name=".SecondActivity"
    android:exported="true">
</activity>
```

```
textView.setOnClickListener {
    val intent = Intent(this, SecondActivity::class.java)
    startActivity(intent)
}
```

기초 UX (2)

```
textView.setOnClickListener {
    val intent = Intent(this, SecondActivity::class.java)
    startActivity(intent)
}
```

Intent란?

안드로이드 앱 컴포넌트 간 통신에 사용되는 객체

- 액티비티 간 전환 (위 예시)
- 푸시 알림 딥링크 (예: 네이버 웹툰 특정 화 푸시알림)
- 공유 / 파일 첨부 등등을 할 때 뜨는 대화창

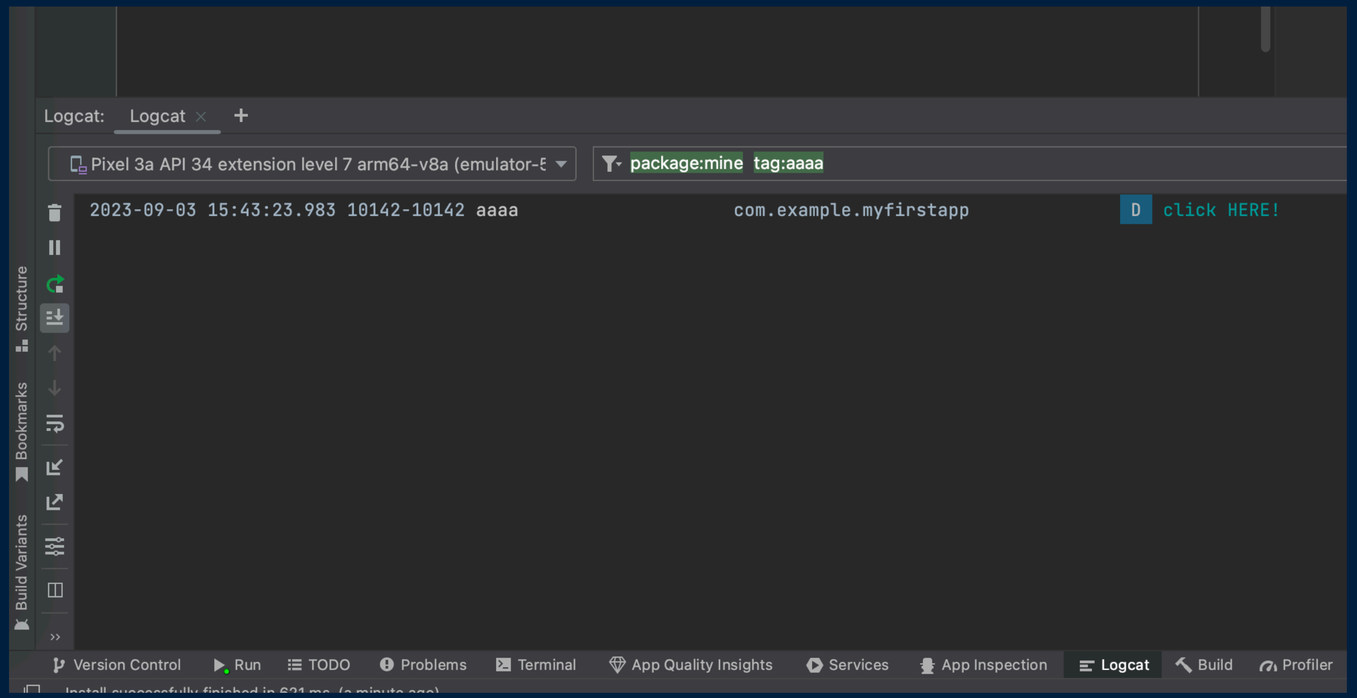
디버깅 하기

Log.d()

오른쪽과 같이 로깅을 하면, Android Studio에서 제공하는 logcat에서 확인할 수 있다.

더 좋은 도구: Timber, Flipper 등등..

```
textView.setOnClickListener {
    Log.d("aaaa", "click HERE!")
}
```



과제, 출결 공지

- 과제 1 기한 : 9월 21일 목 자정 (목~금 넘어가는 밤)

- 과제 링크 : https://github.com/wafflestudio/seminar-2023-android-assignment/tree/main/assignment-1

- 어려울 수 있습니다. 또한 세미나에서 모든 내용을 다룰 수는 없기 때문에 세미나에서 배우지 않은 것도 요구합니다. 하지만 구글링과 ChatGPT, 그리고 세미나장과 조교가 여러분의 곁에 있습니다. 질문해주세요!

- 세미나는 대면 참여가 원칙입니다.

- 3회 이상 결석 시 탈락이며, 지각 2회는 결석 1회입니다. (지각의 기준: 30분 이후 도착)

- 참여하기 어렵거나 비대면으로 참여해야 할 불가피한 사정(훈련소, 예비군, 시험 등 일반적으로 용인되는 것들)이 있을 경우 전날까지 DM을 주시면 결석 횟수에 포함하지 않겠습니다.

- 과제 미제출 혹은 다수 과제가 심각하게 부실할 경우 탈락이며, Grace Day는 모든 과제 통틀어서 4일 제공됩니다.