Kotlin: When to Use Lazy or Lateinit

JEON YONGTAE
https://github.com/yongtaii/yongapps

“Is it safe ?”

JAVA

“Is it safe?” 방대한 코드들이 묻는다.

“Is it safe?” 변수들이 null인지 체크 하도록 강요한다.

“Is it safe?” 이젠 가학적으로 느껴진다.

“It’s so safe you won’t believe it!” 당신은 안전하다고 하지만 확신하진 못한다.

“Is it safe ?” With Kotlin It is

Kotlin

Java는 “billion dollar mistake”로 부터 당신을 지켜주지 못한다. Null Pointer는 모든곳에 숨어 있고, 모든 참조변수는 잠재적으로 Null이 될수 있다.

많은 Android 개발자는 Kotlin에서 답을 찾는다. 현대 개발언어는 보일러플레이트 코드를 줄이고 좀 더 Expressive 한 코드를 추가한다. Kotlin은 Null로부터도 안전하다.

“Is it safe ?” With Kotlin It is

Kotlin

하지만, Java가 왕이었고, 복잡한 Activity LifeCycle에서 Kotlin은 살아 남아야 한다. 예를들어 view를 참조하는 property를 저장하는 방법이 그 중 하나다.

이상적으로, 객체의 property들은 모두 생성과 동시에 정의 된다. 하지만, Activity와 Fragment 객체 생성이 View Loading과 분리되어 있기 때문에, View를 저장하는 property는 초기화 되지 않은 상태로 시작해야 한다.

“Is it safe ?” With Kotlin It is

Kotlin

다음은 View를 참조하는 Property를 다루는 몇몇 접근방법에 대해 소개한다.

- Using a ‘Nullable type’
- Using ‘lateinit’
- Using ‘by lazy’

1 Nullable Type

Nullable Type

Nullable Type

```
class SampleActivity {

    private var sampleAdapter: SampleAdapter? = null

}
```

Property에서 View를 참조하는 가장 간단한 방법은 Nullable Type을 사용하는 것이다.

모든 변수는 반드시 초기화 되어야 하기 때문에, null이 sampleAdapter에 할당된다.

이후에 Activity.onCreate(Fragment.onCreateView) 에서 재할당 될 것이다.

이 때 우리가 원하는 값을 할당 받게 될 것이다.

Nullable Type

Kotlin

```
baseAdapter?.notifyDataSetChanged()
```

Nullable 타입을 사용하게 되면, Nullable 변수에 접근할 때 '?' 혹은 '!!' oerator가 반드시 사용되어야 한다. '?'를 사용하면 sampleAdapter가 null일때 null을 리턴하면서 crash를 방지할 수 있다.

이 것은 아래와 같은 자바코드와 동등하다.

Java

```
if(sampleAdapter != null){
    sampleAdapter.notifyDataSetChanged();
}
```

Nullable Type

Kotlin

```
baseAdapter!!.notifyDataSetChanged()
```

'!!' operator는 baseAdapter가 Null일 경우 크래시를 발생시킬 수 있다.

이런 operator를 사용하는 것은 최소한 baseAdapter가 Nullable인지를 분명히 할 수 있다. Java에서는 이것이 쉽지 않다.

2 lateinit

Nullable Type보다 더 나은 대안책 : Lateinit

lateinit

```
private lateinit var baseAdapter : SampleAdapter
```

'lateinit'을 사용하면, 최초 값 할당은 필요하지 않다.

게다가 baseAdapter가 더 이상 Nullable Type이 아니여도 괜찮다. 따라서, ? 혹은 !! 가 사용되지 않는다.

그러나, 우리는 lateinit var를 사용하기 전에 꼭 할당해 주어야 할 것을 명심해야 한다.

만약 그렇지 않으면 크래시를 발생시킨다.

Nullable Type보다 더 나은 대안책 : Lateinit

example

```
private lateinit var baseAdapter : SampleAdapter

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_k_test)

    baseAdapter = SampleAdapter()
    baseAdapter.count
    baseAdapter.notifyDataSetChanged()

}
```

3 Lazy

lazy

lazy

```
private val baseAdapter : SampleAdapter by lazy {
    SampleAdapter()
}
```

lazy에 의해 정의된 property는 값이 이전에 할당 되었음에도 이를 처음 사용할 때 람다식에 의해 제공된 내용에 의하여 초기화 된다.

lazy 주의사항

lazy

Activity에서 lazy로 초기화된 변수를 setContentView전에 사용할 경우 크래시를 발생한다.

Fragment에서는 View가 onCreateView 내부에서 inflated 된 경우에서도 크래시를 발생할 수 있다. Fragment의 View property가 onCreateView가 완료될 때까지 set되지 않고, lazy 변수의 초기화에서 계속 참조 되기 때문이다.

onViewCreated에서 lazy property를 사용하는 것은 가능하다.

Retained 된 Fragment에서 lazy로 초기화된 property를 사용하면, old View를 계속 참조하고 있기 때문에 메모리 누수가 발생할 수 있다.

4 When to Use Lazy or Lateinit

When to Use Lazy or Lateinit : Lazy

Lazy는 접근 할지 안할지 모르는 Property에 사용하는 것이 좋다. 해당 property에 접근하지 않는다면 초기화 계산을 피할 수 있다.

in Activity

Activity에서 setContentView가 호출되기 전에 접근하지 않는 이상 사용 할 수 있다.

in Fragment

View를 참조하는 Fragment내에서는 적합하지 않다.

onCreateView에서 뷰를 구성하는 일반적인 패턴은 크래시를 유발할 수 있다.

onViewCreated에서 뷰 구성이 사용된다면, 사용해도 괜찮다.

(onViewCreated : onCreateView에서 return해준 view를 가짐.)

When to Use Lazy or Lateinit : Lateinit

in Activity or Fragment

Activity 나 Fragment에서 특히 View를 참조하는 Property 를 상용할 때는 lateinit을 사용하는 것이 타당하다.

우리가 생명주기를 통제하지 않지만, 우리는 property들이 적절히 초기화 될 것을 알고 있다.

단점으로는 적절한 생명주기 메서드안에서 초기화 해야 한다는 것이다.

Referrenced by

- https://www.bignerdranch.com/blog/kotlin-when-to-use-lazy-or-lateinit/

Thanks!

Any questions ?

You can find me at

- jeonyt89@gmail.com