Kotlin lateinit 和 by lazy

buchuqi2677 关注

6 2017.08.02 17:09:30 字数 273 阅读 70,919

lateinit 和 lazy 是 Kotlin 中的两种不同的延迟初始化的实现

lateinit 只用于变量 var，而 lazy 只用于常量 val

> **lazy 应用于单例模式(if-null-then-init-else-return)，而且当且仅当变量被第一次调用的时候，委托方法才会执行。**

`lazy()` 是接受一个 lambda 并返回一个 `Lazy <T>` 实例的函数，返回的实例可以作为实现延迟属性的委托： 第一次调用 `get()` 会执行已传递给 `lazy()` 的 lambda 表达式并记录结果， 后续调用 `get()` 只是返回记录的结果。

```
val lazyValue: String by lazy {
    println("computed!")
    "Hello"
}

fun main(args: Array<String>) {
    println(lazyValue)
    println(lazyValue)
}

打印结果
computed!
Hello

Hello
```

比如这样的常见操作，只获取，不赋值，并且多次使用的对象

```

   private val mUserMannager: UserMannager by lazy {
        UserMannager.getInstance()
    }
```

再比如acitivity中控件初始化的操作，一般传统的进入界面就初始化所有的控件，而使用懒加载，只有用到时才会对控件初始化

```
//kotlin 封装:
fun <V : View> Activity.bindView(id: Int): Lazy<V> = lazy {
    viewFinder(id) as V
}

//acitivity中扩展调用
private val Activity.viewFinder: Activity.(Int) -> View?
    get() = { findViewById(it) }


//在activity中的使用姿势
val mTextView by bindView<TextView>(R.id.text_view)
mTextView.text="执行到我时，才会进行控件初始化"

```

> **lateinit 则用于只能生命周期流程中进行获取或者初始化的变量，比如 Android 的 onCreate()**

```
    @Inject
    @field:Named("home")
    lateinit var pagerAdapter:FragmentStatePagerAdapter
```

再比如

```
class App : Application() {
    init {
        instance = this
    }

    @Inject lateinit var apiComponent: ApiComponent
    override fun onCreate() {
        super.onCreate()
        DaggerApiComponent.builder().apiModule(ApiModule()).appModule(AppModule(this))
    }

    companion object {
        lateinit var instance: App

    }
}
```