

技术小黑屋

- [首页](#)
- [干货铺子](#)
- [存档](#)
- [读书](#)
- [关于](#)
- [合作](#)
- [捐赠](#)
- [订阅](#)
- [好基友](#)
- [内推](#)
- [福利](#)
- [友链](#)

撸了今年阿里、头条和美团的面试，我有一个重要发现.....

编写地道的 Kotlin 代码

May 19th, 2019

Kotlin 以其简洁实用的语法，赢得了很多Java 开发者，尤其是 Android 开发者的喜爱与应用。然而，虽然我们使用 Kotlin 进行编码，可能并没有书写出地道的 Kotlin 代码，亦或者是遵照写Java的思维，用 Kotlin的语法 来编码。

本文将通过多出代码示例，分为Do not（不建议）和Do（建议）两部分，分别代表着不太好的实现和推荐的实现方式，来展示地道的 Kotlin 编码方式。

进行非null判断

```
1 //Do not
2 fun dumpBook(book: Book?) {
3     if (book != null) {
4         book.dumpContent()
5     }
6 }
7
8 //Do
9 fun dumpBook1(book: Book?) {
```



```

10     book?.dumpContent()
11 }

```

进行类型转换并访问一些属性

```

1 // avoid if type checks
2 //Do not
3 fun testTypeCheck(any: Any) {
4     if (any is Book) {
5         println(any.isbn)
6     }
7 }
8
9 //Do
10 fun testTypeCheck0(any: Any) {
11     (any as? Book)?.let {
12         println(it.isbn)
13     }
14 }

```

避免使用!!非空断言

```

1 //Do not
2 fun testNotNullAssertion(feed: Feed) {
3     feed.feedItemList.first().author!!.title
4 }
5
6 //Do
7 fun testNotNullAssertion0(feed: Feed) {
8     feed.feedItemList.first().author?.title ?: "fallback_author_title"
9 }

```

补充：

- 使用!!断言，一旦断言条件出错，会发生运行时异常。

判断可能为null的boolean值

```

1 // Do not
2 fun consumeNullableBoolean() {
3     var isOK: Boolean? = null
4     if (isOK != null && isOK) {
5         //do something
6     }
7 }
8
9
10 //Do
11 fun consumeNullableBoolean0() {
12     var isOK: Boolean? = null
13     if (isOK == true) {
14         //do something
15     }
16 }

```



```

15     }
16 }

```

利用if-else,when,try-catch 的返回值

```

1 //Do not
2 fun testIfElse(success: Boolean) {
3     var message: String
4     if (success) {
5         message = "恭喜 · 成功了"
6     } else {
7         message = "再接再厉"
8     }
9     println(message)
10 }
11
12 //Do
13 fun testIfElse1(success: Boolean) {
14     val message = if (success) {
15         "恭喜 · 成功了"
16     } else {
17         "再接再厉"
18     }
19 }
20
21
22 //Do
23 fun testWhen0(type: Int) {
24     val typeString = when(type) {
25         1 -> "post"
26         2 -> "status"
27         else -> "page"
28     }
29     //can't reassign value to typeString
30 }
31
32 fun getWebContent(url: String): String = TODO()
33
34 //Do
35 fun testTryCatch() {
36     val content = try {
37         getWebContent("https://droidyue.com")
38     } catch (e: IOException) {
39         null
40     }
41     //can't reassign value to content
42 }

```

善用 apply/also/with

```

1 //Do not
2 fun composeIntent(): Intent {
3     val intent = Intent(Intent.ACTION_VIEW)
4     intent.data = Uri.parse("https://droidyue.com")
5     intent.`package` = "com.android.chrome"
6     intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)

```



```
7     return intent
8 }
9
10 //Do
11 fun composeIntent1(): Intent {
12     return Intent(Intent.ACTION_VIEW).apply {
13         data = Uri.parse("https://droidyue.com")
14         `package` = "com.android.chrome"
15         addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
16     }
17 }
```

```
1 data class Request(val uri: String)
2 //use also
3
4 //Do not
5 fun handleRequest(request: Request) : Boolean {
6     return when {
7         request.uri.startsWith("https") -> {
8             handleHttpRequest(request)
9             true
10        }
11
12        request.uri.startsWith("http") -> {
13            handleHttpRequest(request)
14            true
15        }
16
17        else -> false
18    }
19 }
20
21 //Do
22 fun handleRequest1(request: Request): Boolean {
23     return when {
24         request.uri.startsWith("https") -> true.also {
25             handleHttpRequest(request)
26         }
27
28         request.uri.startsWith("http") -> true.also {
29             handleHttpRequest(request)
30         }
31
32         else -> false
33     }
34 }
```

```
1 class Navigator {
2     fun turnLeft() = Unit
3     fun turnRight() = Unit
4     fun forward() = Unit
5     fun backward() = Unit
6 }
7
8 //use with
9 //Do not
10 fun navigate(navigator: Navigator) {
11     navigator.forward()
12     navigator.turnRight()
13     navigator.backward()
14 }
```



```

14     navigator.turnLeft()
15 }
16
17 //Do
18 fun navigate1(navigator: Navigator) {
19     with(navigator) {
20         forward()
21         turnRight()
22         backward()
23         turnLeft()
24     }
25 }

```

直接使用top-level方法，而不是Object里的方法

```

1 //Do not
2 object AppUtil {
3     fun isEnabled(packageName: String): Boolean {
4         TODO()
5     }
6 }
7
8 //Do
9 //AppUtil.kt file
10 fun isEnabled(packageName: String): Boolean {
11     TODO()
12 }

```

使用Kotlin的默认参数特性，而不是方法重载

```

1 //Do not
2 class BadPizza {
3     constructor(size: Float)
4
5     constructor(size: Float, hasCheese: Boolean)
6
7     constructor(size: Float, hasCheese: Boolean, hasBacon: Boolean)
8 }
9
10 //Do
11 class GoodPizza {
12     constructor(size: Float, hasCheese: Boolean = false, hasBacon: Boolean = false)
13 }

```

优先定义并使用扩展方法，而不是Util方法

```

1 //Do not
2 fun isStringPhoneNumber(value: String): Boolean {
3     TODO()
4 }
5
6 //Do
7 fun String.isPhoneNumber(): Boolean = TODO()

```



使用方法引用

```
1 //Do not
2
3 data class NewsItem(val content: String, val isFake: Boolean)
4
5 fun normalLambda() {
6     arrayOf<NewsItem>().filter { it.isFake }.let { print(it) }
7 }
8
9 fun methodReference() {
10     arrayOf<NewsItem>().filter(NewsItem::isFake).let(::print)
11 }
```

使用inline修饰高阶函数（参数为函数时）

```
1 //Do not
2 fun safeRun(block: () -> Unit) {
3     try {
4         block()
5     } catch (t: Throwable) {
6         t.printStackTrace()
7     }
8 }
9 //Do
10 inline fun safeRun0(block: () -> Unit) {
11     try {
12         block()
13     } catch (t: Throwable) {
14         t.printStackTrace()
15     }
16 }
```

备注：

- 关于inline的问题，可以参考[Kotlin 中的 Lambda 与 Inline](#)

把函数参数尽可能放到最后

```
1 //Do not
2 fun delayTask(task: () -> Unit, delayInMillSecond: Long) {
3     TODO()
4 }
5
6 //Do
7 fun delayTask0(delayInMillSecond: Long, task: () -> Unit) {
8     TODO()
9 }
10
11 fun testDelayTasks() {
12     delayTask({
13         println("printing")
14     }, 1000)
15 }
```



```
14     }, 5000L)
15
16     delayTask0(5000L) {
17         println("printing")
18     }
19 }
```

使用mapNotNull

```
1 //Do not
2 fun testMapNotNull(list: List<FeedItem>) {
3     list.map { it.author }.filterNotNull()
4 }
5
6 //Do
7 fun testMapNotNull0(list: List<FeedItem>) {
8     list.mapNotNull { it.author }
9 }
```

尽可能使用只读集合

```
1 fun parseArguments(arguments: Map<String, String>) {
2     //do some bad things
3     //try to clear if the argument is available to be cleared.
4     (arguments as? HashMap)?.clear()
5 }
6
7 //use read-only collections as much as possible
8 //Do not
9 fun useMutableCollections() {
10     val arguments = hashMapOf<String, String>()
11     arguments["key"] = "value"
12     parseArguments(arguments)
13 }
14
15 //Do
16 fun useReadOnlyCollections() {
17     val arguments = mapOf("key" to "value")
18     parseArguments(arguments)
19 }
```

适宜情况下使用Pair或Triple

```
1 // Use Pair or Triple
2 fun returnValues(): Pair<Int, String> {
3     return Pair(404, "File Not Found")
4 }
5
6 fun returnTriple(): Triple<String, String, String> {
7     return Triple("6时", "6分", "60秒")
8 }
```



使用lazy 替代繁琐的延迟初始化

```

1 data class Config(val host: String, val port: Int)
2
3 fun loadConfigFromFile(): Config = TODO()
4
5 //Do not
6 object ConfigManager {
7     var config: Config? = null
8
9     fun getConfig0() : Config? {
10         if (config == null) {
11             config = loadConfigFromFile()
12         }
13         return config
14     }
15 }
16
17 //Do
18 object ConfigManager1 {
19     val config: Config by lazy {
20         loadConfigFromFile()
21     }
22 }

```

使用lateinit 处理无法再构造函数初始化的变量

```

1 //Do not
2 class FeedItem {
3     var author: Feed.Author? = null
4 }
5
6 //Do
7 class FeedItem0 {
8     lateinit var author: Feed.Author
9 }

```

善用Data class的copy方法

```

1 //Do not
2 class Car {
3     private var engine: String? = null
4
5     constructor(theEngine: String) {
6         engine = theEngine
7     }
8
9     constructor(car: Car) {
10         engine = car.engine
11     }
12 }
13
14 //Do
15 data class Car0(val engine: String)

```




```

16
17
18 fun test() {
19     val firstCar = Car("Honda")
20     val secondCar = Car(firstCar)
21
22     val thirdCar = Car0("Nissan")
23     val fourthCar = thirdCar.copy()
24     val fifthCar = thirdCar.copy(engine = "Ford")
25 }

```

针对函数类型和集合使用typealias

```

1 //Do not
2
3 interface OnValueChangedListener {
4     fun onValueChanged(value: String)
5 }
6
7 //Do
8 typealias OnValueChangedListener0 = (String) -> Unit
9
10 val value : OnValueChangedListener0 = {
11     println(it)
12 }
13
14 //Do
15 typealias BookSet = HashSet<Book>
16
17 val bookSet = BookSet().apply {
18     add(Book("978-0131872486"))
19 }

```

使用含义更加清晰的substringBefore和substringAfter

```

1 //Do not
2 fun testSubstring() {
3     val message = "user|password"
4     Log.i("testSubstring.user=", message.substring(0, message.indexOf("|")))
5
6     Log.i("testSubstring.password=", message.substring(message.indexOf("|") + 1))
7 }
8
9
10 fun testSubstring0() {
11     val message = "user|password"
12     Log.i("testSubstring.user=", message.substringBefore("|"))
13
14     Log.i("testSubstring.password=", message.substringAfter("|"))
15 }

```

以上就是一些相对更加Kotlin style的代码示例，如有补充，请在下方评论指出。谢谢。

相关阅读



- [研究学习Kotlin的一些方法](#)
- [Kotlin 中的 Lambda 与 inline](#)
- [有点意思的Kotlin的默认参数与JVMOverloads](#)

《Kotlin核心编程》



长按识别二维码购买

作者：水滴技术团队


适读人群：本书面向的读者需具备一定的 Java 或 Kotlin 编程经验，对 Kotlin、Android 开发或函数式编程感兴趣，处于中级认知水平。

(1) 本书作者团队强大：水滴技术团队出品，团队核心成员均来自阿里、淘宝，其中不乏章建良（网名：Jilen）等技术大牛。

(2) 书中干货多，且具备一定深度：本书不是一般泛泛讲语法的书，而是一般围绕Kotlin设计理念对所有核心内容进行详细剖析的书，有一点Java基础，想高层次入门Kotlin的读者，本所不容错过。

Posted by androidyue May 19th, 2019 [Java](#), [Kotlin](#)[« Kotlin 和 Flutter 对于开发者究竟意味着什么？送书啦 · Kotlin · Android · Flutter都有！ »](#)

优质知识付费



技术小黑屋
我已加入学习，邀你一起！



设计模式之美

前 Google 工程师手把手教你写高质量代码

你将获得


- 23 种设计模式与范式实战精讲
- 200+ 真实案例分析设计与实现
- 顶尖互联网公司的编程经验分享
- 应对设计模式面试的思路与技巧

王争

前 Google 工程师
《数据结构与算法之美》专栏作者



原价¥199
早鸟优惠 ¥129
学生特权，立享 5 折





技术小黑屋
我已加入学习，邀你一起！



左耳朵耗子

全年独家专栏

你将获得

- 主流语言编程范式详解
- 分布式系统关键技术
- 微服务与性能调优实战经验
- 程序员练级攻略

陈皓

资深技术专家
骨灰级程序员



¥299 / 104期全集
新人立减 ¥30



0 筆評論

訪客 ▾



寫點什麼

① 支援 Markdown 語法

使用 GitHub 登入



成為首個留言的人吧！

最新文章

- [终端依赖者福利：终端也能实现翻译功能了](#)
- [Mac 下在终端直接查看图片](#)
- [在 Android 中如何确定 App\(Activity\) 的启动者](#)
- [这可能是最好的 Android/Kotlin 日志输出方法](#)
- [简简单单对比 encodeURI 与 encodeURIComponent](#)
- [GapWorker 导致 RecyclerView 视频播放声音残留问题](#)
- [在终端使用脚本查看网站 SSL 证书信息](#)
- [Android WebView 诊断与排查问题的方法和技巧](#)
- [使用脚本批量替换文本内容](#)
- [一个小技巧提升 OkHttp 请求稳定性](#)
- [根据包名查找 Jar 包文件](#)
- [解决 Grep 的多次管道过滤问题](#)
- [JvmMultifile 注解在 Kotlin 中的应用](#)
- [JvmName 注解在 Kotlin 中的应用](#)
- [Contract · 开发者和 Kotlin 编译器之间的契约](#)
- [简单几招提速 Kotlin Kapt 编译](#)
- [Kotlin 编译与 Intrinsics 检查](#)
- [Kotlin 编译调校之 WarningsAsErrors](#)
- [使用 Kotlin Reified 让泛型更简单安全](#)
- [为 Kotlin 项目设置编译选项](#)

Blogrolls

- [技术小黑屋血拼](#)
- [OneDrive 超实用网盘](#)
- [通勤读书神器 Kindle](#)
- [XBrowser-极简快速的浏览器](#)
- [Digital Ocean VPS：注册赠送50刀](#)
- [玩客云 私人云盘 迅雷 畅快下载 轻NAS](#)
- [Github程序员,极客IT男女秋冬装连帽卫衣外套](#)
- [Vultr VPS, 注册赠送50刀 · 节点丰富,最低3美刀/月](#)
- [超强路由器 网件NETGEAR WIFI双频千兆无线路由器 中继 OPENWRT多拨](#)

人气文章

- [程序员必读的六本书](#)
- [Java程序员必读的9本书](#)
- [纠结才能写出好代码](#)
- [程序员的逗比瞬间（第一季）](#)
- [探究 android:largeHeap](#)
- [这就是观察者模式](#)
- [说说依赖注入](#)



- [Android中Java和JavaScript交互最详细的文章](#)
- [土别三年，我的最新个人作品](#)
- [十分钟掌握SQLite操作](#)

扫一扫关注小黑屋公众号(droidyue_com)



Copyright © 2019 - androidyue - Powered by [Octopress](#) ; 优质服务品牌 : [OneDrive](#) [Vultr VPS](#) [Digital Ocean](#) [阿里云](#)

