Kotlin：简化版的Scala

2019-07-06

行走江湖的剑客，必然要有一柄趁手的宝剑。好的程序语言就像一把好剑，重量合适，拿着舒服，挥舞起来优雅，杀伤力过关。Kotlin官方对待Kotlin和Scala的关系是，“如果你玩Scala很happy，那你就不需要Kotlin。”

脚本化

Scala执行的基本单位和Java一样是类，而Kotlin允许文件中的main方法直接运行，不需要类。Java的入口函数定义在类中：

```
public class Java {
    public static void main(String[] args) {}
}
```

Scala的入口函数定义在样本类而不是普通的类中：

```
object Scala {
  def main(args: Array[String]): Unit = {}
}
```

Kotlin的入口函数则直接定义在.kt文件中，相应的，Kotlin的类仅相当于一种数据结构，类中无法定义入口函数：

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

构造函数与单例模式

Kotlin的构造函数同Scala一样写在类定义处，因此也无法像Java的构造函数一样直接写入初始化代码。Kotlin中使用init代码块来执行初始化程序：

```
class Test(arg: String) {
    init {
        println("This string is ${arg}")
    }
}

fun main(args: Array<String>) {
    val test = Test("smallyu")
}

// This string is smallyu
```

如果需要第二个构造函数，就要使用类似ES6的constructor函数，或者类似Scala的辅助构造器。这实在是丑陋的写法，相比之下Java真的友善多了。

```
class Test(arg1: String) {
    init {
        println("This string is ${arg1}")
    }
    constructor(arg2: Int): this("smallyu2") {
        println("This int is ${arg2}")
    }
}

fun main(args: Array<String>) {
    val test = Test(1)
}

// This string is smallyu2
```

```
// This int is 1
```

Kotlin的构造函数是需要用constructor关键字定义的，默认可以省略，但如果要加权限修饰符自然就不能省了。在Kotlin中实现单例模式的思路与Java相同，让构造器私有，然后通过静态方法暴露实例：

```
class Test private constructor() {
    companion object Factory {
        fun create(): Test = Test()
    }
}

fun main(args: Array<String>) {
    val test = Test.Factory.create()
}
```

Kotlin中的object定义静态代码块，companion允许在类内部定义静态代码块，因此compaion object定义了类外部可以访问的方法create()。

getter和setter

Kotlin另一个有趣的玩意儿是getter和setter。前端框架React或Vue实现数据双向绑定的原理即使用Object.defineProperty()定义对象的getter和setter，使得对象的变化可以实时同步到页面上。Kotlin提供了对属性getter和setter的支持：

```
var test: Int
    get() {
        println("There is test getter")
        return 2
    }
    set(arg) {
        println("The setter arg is ${arg}")
    }

fun main(args: Array<String>) {
    println(test)
    test = 3
}

// There is test getter
// 2
// The setter arg is 3
```

其他

开始对Kotlin感兴趣是因为发现Kotlin竟然支持协程，如果Kotlin真的有语言级别的协程支持，加上运行在Jvm上的特点，以及能够开发多平台应用包括Server Side、Android、JavaScript、Native，那Kotlin无疑是异常强大的编程语言。然而事实上Kotlin的协程只是一个扩展包，甚至还需要使用编译工具来引入，对协程的支持还是Go语言独大。用于JavaScript平台也是个幌子，并没有比TypeScript好用，至于Android和Native本身也是Java的应用场景……

Kotlin提供了许多语法糖，看似可以简化程序员的代码量，但是为了熟练应用Kotlin的特性，使用者又不得不搞清楚类似data class的概念，就像Scala的case class一样。Kotlin的学术性弱于Scala，工程能力又不比Java有大的优势。Go语言虽然另辟蹊径，语言特性上有广为诟病的地方，但是看着爽，写着也爽。所以Kotlin和Scala一样，并不会有广泛的应用前景。也就是说，它并不会是下一个很流行的编程语言。