Introdução ao Kotlin para Desenvolvimento Mobile com Android

Pablo Leon Rodrigues

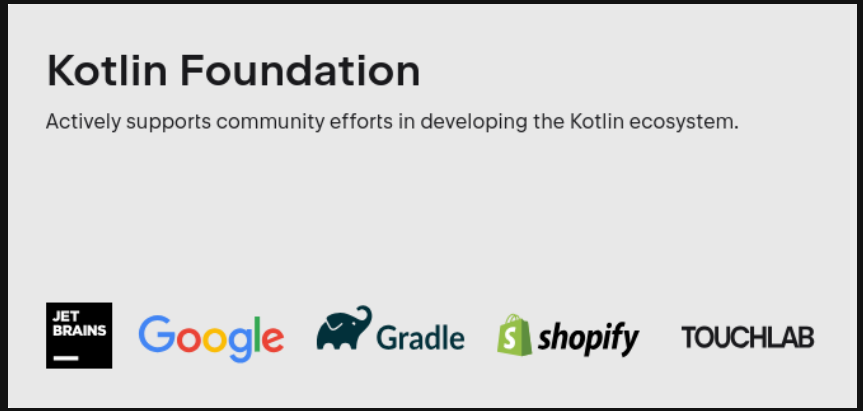
1. Tópicos Especiais em Desenvolvimento de Software II
    1. Kotlin
    2. Hello, world!

Kotlin

Kotlin é uma linguagem de programação multiplataforma, orientada a objetos e funcional, concisa e estaticamente tipada, desenvolvida pela JetBrains em 2011, que compila para a Máquina virtual Java e que também pode ser traduzida para a linguagem JavaScript e compilada para código nativo.

Uma das características mais importantes do Kotlin é a interoperabilidade fluida com Java. Como o código Kotlin é compilado até o bytecode da JVM, seu código Kotlin pode ser chamado diretamente no código Java e vice versa. Isso significa que você pode aproveitar bibliotecas Java já existentes diretamente do Kotlin.

Além disso, a maioria das APIs do Android é gravada em Java, e você pode chamá-las diretamente do Kotlin.



Hello, world!

```
fun main() {
    println("Hello, world!")
}
```

`fun` é usado para declarar funções

a função `main()` é onde o programa inicia

o corpo da função é escrito entre chaves `{}`

`println()` e `print()` são usadas como saída padrão

`;` não é necessário!

Simple

```
fun main() {
    val name = "stranger"       // Declare your first variable
    println("Hi, $name!")       //  ...and use it!
    print("Current count:")
    for (i in 0..10) {          // Loop over a range from 0 to 10
        print(" $i")
    }
}
```

Result

```
Hi, stranger!
Current count: 0 1 2 3 4 5 6 7 8 9 10
```

Asynchronous

```
import kotlinx.coroutines.*

suspend fun main() {                              // A function that can be suspended and resumed later
    val start = System.currentTimeMillis()
    coroutineScope {                              // Create a scope for starting coroutines
        for (i in 1..10) {
            launch {                              // Start 10 concurrent tasks
                delay(3000L - i * 300)            // Pause their execution
                log(start, "Countdown: $i")
            }
        }
    }
    // Execution continues when all coroutines in the scope have finished
    log(start, "Liftoff!")
}

fun log(start: Long, msg: String) {
    println("$msg " +
            "(on ${Thread.currentThread().name}) " +
            "after ${(System.currentTimeMillis() - start)/1000F}s")
}
```

Object-Oriented

```
abstract class Person(val name: String) {
    abstract fun greet()
}

interface FoodConsumer {
    fun eat()
    fun pay(amount: Int) = println("Delicious! Here's $amount bucks!")
}

class RestaurantCustomer(name: String, val dish: String) : Person(name), FoodConsumer {
    fun order() = println("$dish, please!")
    override fun eat() = println("*Eats $dish*")
    override fun greet() = println("It's me, $name.")
}

fun main() {
    val sam = RestaurantCustomer("Sam", "Mixed salad")
    sam.greet() // Implementation of abstract function
    sam.order() // member function
    sam.eat()   // Implementation of interface function
    sam.pay(10) // Default implementation in interface
}
```

Functional

```
fun main() {
    // Who sent the most messages?
    val frequentSender = messages
        .groupBy(Message::sender)
        .maxByOrNull { (_, messages) → messages.size }
        ?.key                                          // Get their names
    println(frequentSender) // [Ma]

    val senders = messages
        .asSequence()                                  // Make operations lazy (for a long call chain)
        .filter { it.body.isNotBlank() && !it.isRead }  // Use lambdas...
        .map(Message::sender)                          //  ...or member references
        .distinct().sorted().toList()                  // Convert sequence back to a list to get a result
    println(senders) // [Adam, Ma]
}

data class Message(                                    // Create a data class
    val sender: String,
    val body: String,
    val isRead: Boolean = false,                       // Provide a default value for the argument
)
val messages = listOf(                                 // Create a list
    Message("Ma", "Hey! Where are you?"),
    Message("Adam", "Everything going according to plan today?"),
    Message("Ma", "Please reply. I've lost you!"),
)
```

Tests Integrated

```
import org.junit.Test                    // Tests
import kotlin.test.*                     // The following example works for JVM only

class SampleTest {
    @Test
    fun `test sum`() {                   // Write test names with whitespaces in backticks
        val a = 1
        val b = 41
        assertEquals(42, sum(a, b), "Wrong result for sum($a, $b)")
    }

    @Test
    fun `test computation`() {
        assertTrue("Computation failed") {
            setup()                      // Use lambda returning the test subject
            compute()
        }
    }
}

fun sum(a: Int, b: Int) = a + b          // Sources
fun setup() {}
fun compute() = true
```

https://www.jetbrains.com/

https://www.jetbrains.com/pt-br/