链滴

怎么骗你的测试女朋友用 java 写测试用例

作者：crick77

原文链接：https://ld246.com/article/1600503758989

来源网站：链滴

首先你要有个女朋友

其次要回答3个灵魂拷问

1. 使用成本
2. 能节省什么
3. 能带来什么

最好是感知不到是在写代码。

实操

上篇《距离全栈 你只差一个kotlinx》聊到了

Kotlin 借助 Lambda + Extensions扩展 来实现内部DSL，这次展开聊聊 扩展和 操作符重载是如何让的代码不像代码的。

扩展

Kotlin 可以对一个类的属性和方法进行扩展，且不需要继承或使用 Decorator 模式。

扩展是一种静态行为，对被扩展的类代码本身不会造成任何影响。

扩展方法

如果我希望对一个已有类增加一个方法，就可以直接给这个类增加扩展方法，同时所有的对象都可以用这个扩展方法。

举例说明

```
fun String.around(wrap: String): String {
    return "$wrap$this$wrap"
}

fun String.around(left: String, right: String): String {
    return "$left$this$right"
}

@Test
fun length_should_return_string_concat_around() {
    assertEquals("""
        ===a===
        ===b===
        ===c===
    """.trimIndent(), arrayListOf("a", "b", "c").joinToString("\n") { it.around("===") })
    assertEquals("<a>", "a".around("<", ">"))
}
```

扩展属性

扩展属性与成员变量不同，因为扩展不会真的在类中插入一个成员变量。因此扩展属性不能初始化，能通过getter方法定义

```
fun String.获取长度(): Int = this.length

val String.长度: Int = this.length //编译错误
val String.长度: Int = 1 //编译错误

val String.长度: Int
    get() = this.length

@Test
fun length_should_return_size_of_string() {
    assertEquals("abc".length, 3)
    assertEquals("abc".长度, 3)
    assertEquals("abc".获取长度(), 3)
}
```

操作符重载

Kotlin允许我们为自己的类型提供预定义的一组操作符实现（这些操作符都对应的成员函数或扩展函），他们是一一对应的

算术运算符operator

基本类型中，Int + Int 表示数字相加， String + String 则代表了字符串拼接，那么如果是 ArrayList<String> + String 呢?

kotlin中重载了Collection的算术运算符，其中 plus 对应了 +的操作函数。所以 ArrayList<String> + String 会把 String 添加到 ArrayList<String>集合中

```
/**
 * Returns a list containing all elements of the original collection and then the given [element].
 */
public operator fun <T> Collection<T>.plus(element: T): List<T> {
    val result = ArrayList<T>(size + 1)
    result.addAll(this)
    result.add(element)
    return result
}
```

我们也可以自定义 +的重载，比如将 String拼接到 ArrayList<String>每个元素的末尾

```
operator fun ArrayList<String>.plus(element: String): List<String> {
    return this.map { "$it$element" }
}

@Test
fun list_plus() {
    val list: ArrayList<String> = arrayListOf("a", "b", "c")
    assertEquals(arrayListOf("a1", "b1", "c1"), list + "1")
}
```

看起来和扩展函数有些类似，我们继续放飞

中缀表示法infix

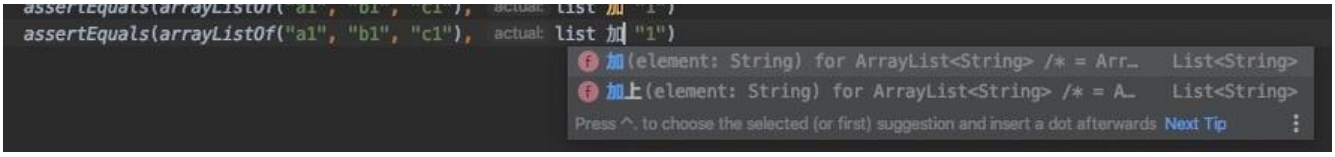
标有 infix 关键字的函数也可以使用中缀表示法（忽略该调用的点与圆括号）调用。中缀函数必须满以下要求：

- 它们必须是成员函数或扩展函数；
- 它们必须只有一个参数；
- 其参数不得接受可变数量的参数且不能有默认值。

通过这种表示方法，我们的代码逻辑可以变成陈述句

```
infix fun Int.加(x: Int): Int {
    return this + x
}

infix fun ArrayList<String>.加(element: String): List<String>  {
    return this.map { "$it$element" }
}

infix fun Any.等于(x: Any): Boolean {
    return this == x
}

@Test
fun list_plus() {
    assertTrue( arrayListOf("a", "b", "c") 加 "1" 等于 arrayListOf("a1", "b1", "c1"))
}

@Test
fun advanced_mathematics() {
    assertTrue( 1 加 1 等于 2)
}
```

并且有IDE的高亮及语法提示支持



这条用例的代码元素还有2个，JUnit + fun

调用操作符invoke

调用操作符

圆括号转换为调用带有适当数量参数的 invoke。

表达式	翻译为
a()	a.invoke()
a(i)	a.invoke(i)
a(i, j)	a.invoke(i, j)
a(i_1, ……, i_n)	a.invoke(i_1, ……, i_n)

我们可以通过重载 invoke的方法移除 fun的概念，变成 标题 {内容}的格式进行触发函数。 而 JUnit 以通过父类完成TestCase的注册

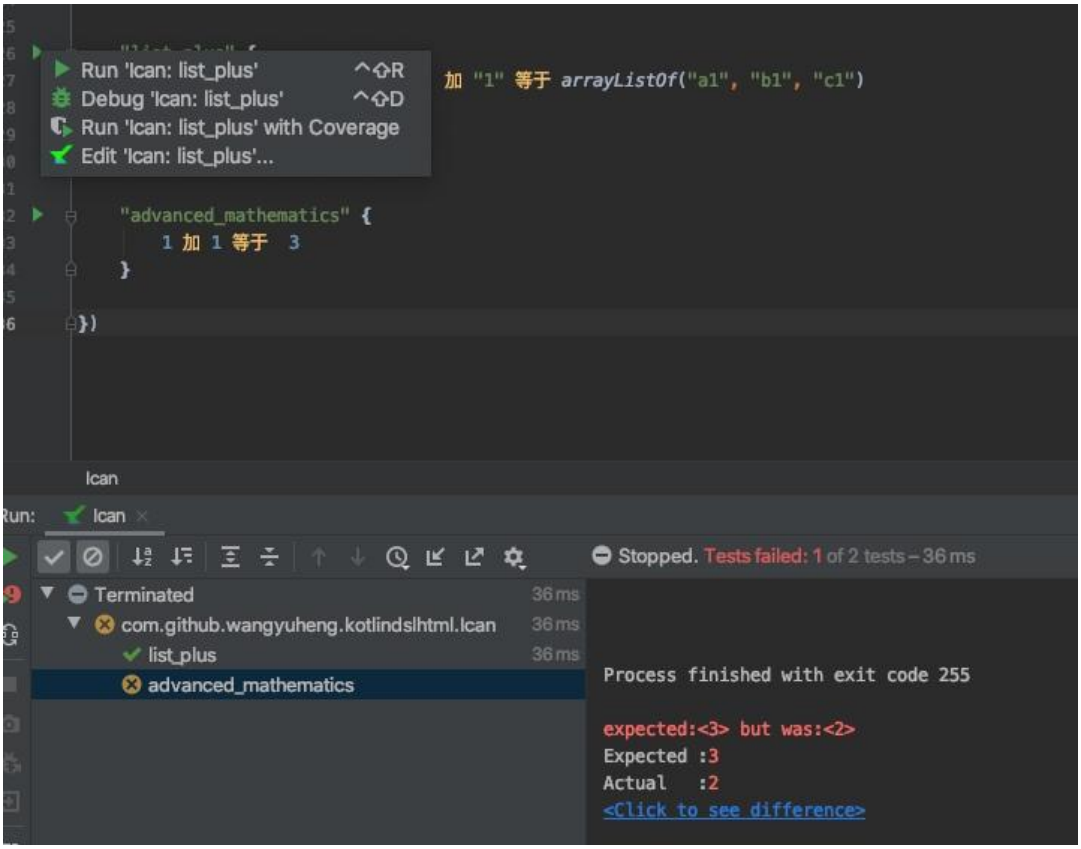
原文链接：怎么骗你的测试女朋友用 java 写测试用例

```kotlin
infix operator fun String.invoke(test: suspend Any.() -> Unit) {
}

abstract class Base(body: Base.() -> Unit = {}) {

    init {
        body()
    }

}

class Ican : Base({

    "list_plus" {
        arrayListOf("a", "b", "c") 加 "1" 等于 arrayListOf("a1", "b1", "c1")
    }
    "advanced_mathematics" {
        1 加 1 等于 2
    }

})
```

kotest

Kotest is a flexible and comprehensive testing project for Kotlin with multiplatform support.

TestCase的注册，以及如何在idea中运行某个测试方法比较复杂，我们可以直接使用 Kotest及其插完成。而 Kotest的实现原理就是上文提到的**扩展**以及**操作符重载**



解答

原文链接：怎么骗你的测试女朋友用 java 写测试用例

看一下是否能回答上述灵魂拷问

1. 使用成本

1. 通过IDE约束及提示，简化编写成本

2. 增加了额外的约束。但是从DDD的过程来看，统一语言在团队协作方面有着重要地位。命名表明你对事物的理解，所以团队内统一描述语言是很有必要的

2. 能节省什么

1. 通过代码和工具函数的封装，可以简化重复操

2. 可以直接调用开发代码方法及接口

3. 通过封装简化接口自动化成本

```
"get_user_info" {
    http {
        url = "http://localhost:8080/user"
        method = "GET"
    }.状态码 等于 200
}
```

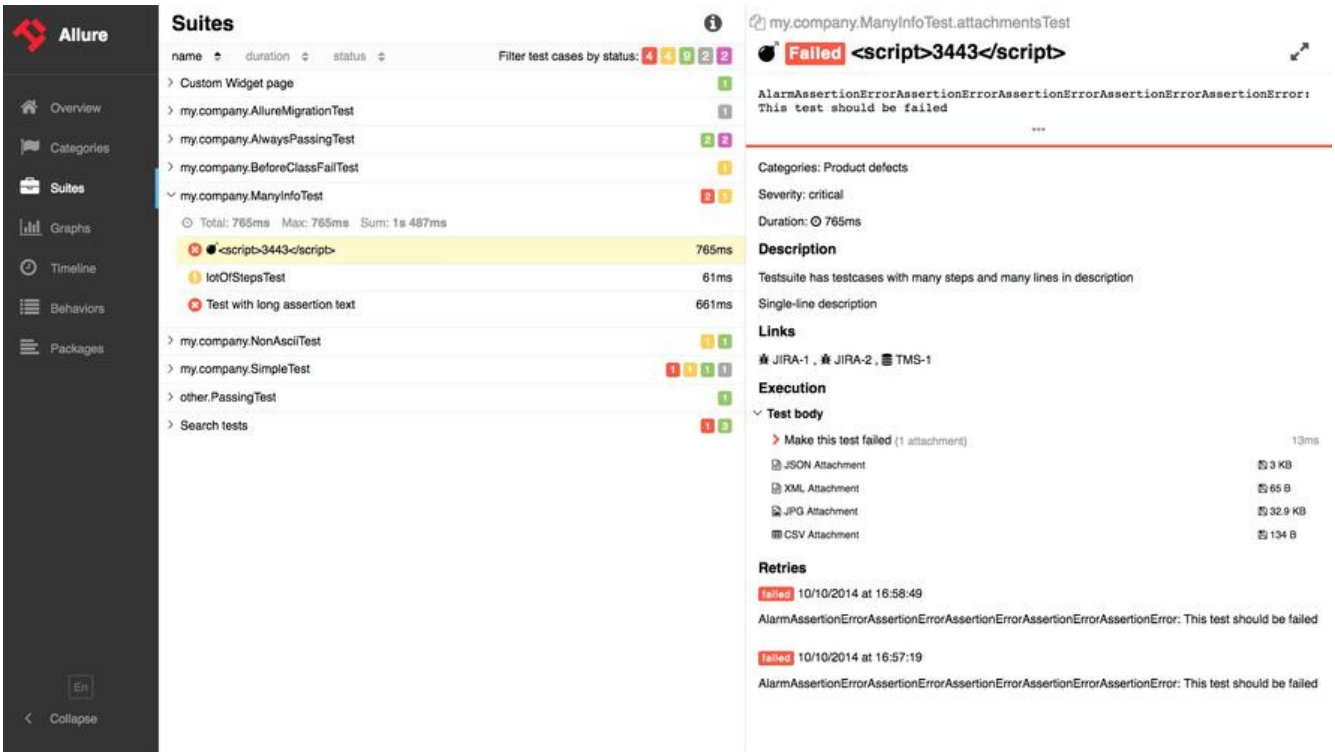
如果看不懂这个语法，可以看上一篇的DSL

3. 能带来什么

1. 配合CICD在潜移默化中完成自动化测试并输出测试报告

2. 借助代码管理工具(gitlab、github)，完成统计、review、操作留痕、版本等管理操作

原文链接：怎么骗你的测试女朋友用 java 写测试用例



结论

1. 为什么标题说java，内容都是kotlin? 因为这篇是标题党
2. 使用扩展or重载时，一定要明确 **函数作用域**
3. 为什么要做扩展和操作符重载? **Because I can**

原文链接：怎么骗你的测试女朋友用 java 写测试用例