Kotlin

# Run,apply,let,also,with的用法和区别

## Run:

* run函数块。返回值为函数块最后一行，或者指定return表达式
* 调用某对象的run函数，在函数块内可以通过 *this* 指代该对象。返回值为函数块的最后一行或指定return表达式

val a = run {

println("run")

return@run 3

}

println(a)

运行结果：

run

3

## Apply

调用某对象的apply函数，在函数块内可以通过 *this* 指代该对象。返回值为该对象自己

## Let

调用某对象的let函数，则该对象为函数的参数。在函数块内可以通过 *it* 指代该对象。返回值为函数块的最后一行或指定return表达式。

## Also

调用某对象的also函数，则该对象为函数的参数。在函数块内可以通过 *it* 指代该对象。返回值为该对象自己

## With

with函数和前面的几个函数使用方式略有不同，因为它不是以扩展的形式存在的。它是将某对象作为函数的参数，在函数块内可以通过 *this* 指代该对象。返回值为函数块的最后一行或指定return表达式。

实例：

val a = with("string") {

println(this) 3

}

println(a)

运行结果：

string

# 双冒号::

Kotlin中将函数作为参数传递，使用双冒号::引用其他方法

# xml中的使用语法

## 每一个变量variable都是由名称name和类型type组成。

name可以在布局文件中使用，也可以通过Setter和Getter在代码中使用。type可以是基本数据类型、集合、适配器、自定义类等，除了基本类型，其他都必须把包名写全

## 双引号中可以套“`”（Tab键上面，非单引号），单引号内可以套双引号；

## 不能直接用boolean和int类型的值。

而且int值将被当做资源文件。因此要使用其数值，必须转换成字符串，如（boolean同理）：   
"@{user.age + ``}"，或 "@{String.valueOf(user.age)}"

## 颜色必须使用ARGB制，指定其所有的透明度和单色值，以前的#F00和#FF0000无效，且前面不能加“#”

## "@{user.nickname ?? user.name}"

代表user.nickname为null时显示user.name，否则显示自己。等同于"@{user.nickname == null ? user.name : user.nickname}"

## 比较运算符，必须使用转义字符

大于号“>”——&gt;   
小于号“<”——&lt;

## 不能直接使用中文（MD），如：： android:text='@{user.male ? "男" : "女"}'  将报错，得用string引用，改成：  android:text='@{user.male ? @string/male : @string/female}'

## 注意事项

### 在type中的泛型同样不能使用”<”与”>”，必须使用转义字符"&lt;"和"&gt;"

### 转义字符在type中会包红色错误，可忽略，编译时能通过

### 不能直接使用android:text="@{userList[0]}"，即使你想打印对象的地址，也不行。这样使用的后果是：每个databing都报“程序包不存在”的错误，连之前正常的也错了。会懵逼的

### 子布局中的(user.male ? `男` : `女`)，必须用小括号括起来

### android:text中的数据组合很长，但也只能写一行，目前还不知如何分行

# kotlin的常用操作符

常用操作符大致可以分为

* 总数操作符
* 过滤操作符
* 映射操作符
* 元素操作符
* 生产操作符
* 顺序操作符

## 总数操作符

**private val list** = *listOf*<Int>(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)  
 *//any 只要有一个符合就返回true* **val any** = **list**.*any* **{ it** > 8 **}** *//all 所有条件符合才返回true* **val all** = **list**.*all* **{ it** > 0 **}** *//count 返回符合条件的数目* **val count** = **list**.*count* **{ it** > 5 **}** *//none 如果没有任何元素与给定的函数匹配，则返回true* **val none** = **list**.*none* **{ it** > 10 **}** *//fold 在一个初始值的基础上 从第一项到最后一项通过 一个函数操作 所有的元素。  
//下面是初始值4 每项进行累加* **val fold** = **list**.*fold*(4) **{** total, next **->** total + next **}** *//foldRight与fold一样，但是顺序是从最后一项到第一项。注意与fold的区别，参数位置调过来了* **val foldRight** = **list**.*foldRight*(4) **{** next, total **->** total + next **}** *//reduce 从第一项到最后一项通过 一个函数操作 所有的元素，相对于fold，没有初始值  
//reduceRight 是从后到前* **val reduce** = **list**.*reduce* **{** acc, i **->** acc + i **}** *//forEach 遍历每个元素并且进行操作* **val foreach** = **list**.*forEach* **{** *println*(**it**) **}** *//forEachIndexed 与foreach相同，但是可以得到index* **val forEachIndexed** = **list**.*forEachIndexed* **{** index, value **->** *println*(**"$**index **-> $**value**"**) **}** *//max 返回最大的值，如果没有则返回null min同* **val max** = **list**.*max*()  
  
 *//maxBy 根据指定的函数返回最大值 minBy同* **val maxBy** = **list**.*maxBy* **{** -**it }** *//sumBy 每项经过函数转换后的和* **val sumBy** = **list**.*sumBy* **{ it** + 9 **}**

## 滤操作符

**private val list** = *listOf*<Int>(0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0)  
  
*/\*\*  
 \* drop 返回包含去掉前n个元素的所有元素的列表  
 \* Returns a list containing all elements except first* ***[n]*** *elements.  
 \* 返回[4, 5, 6, 5, 4, 3, 2, 1, 0]  
 \*/***val drop** = **list**.*drop*(4)  
  
*/\*\*  
 \* dropwhile 根据特定的函数 从第一项开始 直到不满足条件后返回 列表  
 \* Returns a list containing all elements except first elements that satisfy the given* ***[predicate]****.  
 \* 返回[0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0]  
 \*/***val dropwhile** = **list**.*dropWhile* **{ it** > 1 **}***/\*\*  
 \* dropLastWhile 返回根据特定的函数 从最后一项开始 直到不满足条件后返回 列表  
 \* Returns a list containing all elements except last elements that satisfy the given* ***[predicate]****.  
 \* 返回[0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0]  
 \*/***val dropLastWhile** = **list**.*dropLastWhile* **{ it** > 4 **}***/\*\*  
 \*filter 返回所有符合给定函数条件的元素。  
 \* Returns a list containing only elements matching the given* ***[predicate]****.  
 \* [5, 6, 5]  
 \*/***val filter** = **list**.*filter* **{ it** > 4 **}***/\*\*  
 \* filterNot 返回所有不符合给定函数条件的元素  
 \* Returns a list containing all elements not matching the given* ***[predicate]****.  
 \* [0, 1, 2, 3, 4, 4, 3, 2, 1, 0]  
 \*/***val filterNot** = **list**.*filterNot* **{ it** > 4 **}***/\*\*  
 \* filterNotNull 返回非null元素  
 \* Returns a list containing all elements that are not `null`.  
 \* [0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0]  
 \*/***val filterNotNull** = **list**.*filterNotNull*()  
  
*/\*\*  
 \* 返回满足该ranger的元素集合  
 \* Returns a list containing elements at indices in the specified* ***[indices]*** *range.  
 \* [0, 1, 2, 3, 4, 5, 6]  
 \*/***val slice** = **list**.*slice*(0..6)  
  
*/\*\*  
 \* listOf(0,4,7)是集合list的坐标  
 \* Returns a list containing elements at specified* ***[indices]****.  
 \* [0, 4, 5]  
 \*/***val slice2** = **list**.*slice*(*listOf*(0, 4, 7))  
  
*/\*\*  
 \*返回前n项  
 \* Returns a list containing first* ***[n]*** *elements.  
 \* [0, 1, 2, 3]  
 \*/***val take** = **list**.*take*(4)  
  
*/\*\*  
 \* 返回后n项  
 \* Returns a list containing last* ***[n]*** *elements.  
 \* [3, 2, 1, 0]  
 \*/***val takeLast** = **list**.*takeLast*(4)  
  
  
*/\*\*  
 \* 从第一项开始判断，直到不符合就返回，返回符合的前几项数据  
 \* Returns a list containing first elements satisfying the given* ***[predicate]****.  
 \* []  
 \*/***val takeWhile** = **list**.*takeWhile* **{ it** > 4 **}**

## 映射操作符

**private val list** = *listOf*(0, 1, 2, 3, 4, 5, 4, 3, 2, 1, 0, -1)  
  
*/\*\*  
 \* 返回满足条件的集合  
 \* Returns a list containing the results of applying the given* ***[transform]*** *function  
 \* to each element in the original collection.  
 \* [false, false, false, true, true, true, true, true, false, false, false, false]  
 \*/***val map** = **list**.*map* **{ it** > 2 **}***/\*\*  
 \* 返回特定函数后的集合，参数是Iterable类型，  
 \* Returns a single list of all elements yielded from results of* ***[transform]*** *\* function being invoked on each element of original collection.  
 \* [0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 4, 5, 3, 4, 2, 3, 1, 2, 0, 1, -1, 0]  
 \*/***val flatMap** = **list**.*flatMap* **{** *listOf*(**it**, **it** + 1) **}***/\*\*  
 \* 根据函数将集合分组，返回map类型对象  
 \* Groups elements of the original collection by the key returned by the given* ***[keySelector]*** *function  
 \* applied to each element and returns a map where each group key is associated with a list of corresponding elements.  
 \*  
 \* The returned map preserves the entry iteration order of the keys produced from the original collection.  
 \* {false=[0, 1, 2, 3, 3, 2, 1, 0, -1], true=[4, 5, 4]}  
 \*  
 \** ***@sample samples.collections.Collections.Transformations.groupBy*** *\*/***val groupBy** = **list**.*groupBy* **{** value **->** value > 3 **}***/\*\*  
 \* 返回一个集合，通过 角标和值 来生成  
 \* Returns a list containing the results of applying the given* ***[transform]*** *function  
 \* to each element and its index in the original collection.  
 \** ***@param [transform]*** *function that takes the index of an element and the element itself  
 \* and returns the result of the transform applied to the element.  
 \* [0, 1, 2, 3, 4, 5, 4, 3, 2, 1, 0, -1]  
 \*/***val mapIndexed** = **list**.*mapIndexed* **{** index, value **->** value **}***/\*\*  
 \* 返回一个每一个非null元素根据给定的函数转换所组成的List  
 \* Returns a list containing only the non-null results of applying the given* ***[transform]*** *function  
 \* to each element in the original collection.  
 \* [0, 2, 4, 6, 8, 10, 8, 6, 4, 2, 0, -2]  
 \*/***val mapNotNull** = **list**.*mapNotNull* **{ it** \* 2 **}**

## 元素操作符

**private val list**= *listOf*(0,1,2,3,4,5,6,4,3,2,1,0,-1)  
*/\*\*  
 \* 返回给定index对应的元素，如果index数组越界则会根据给定函数返回默认值,第二个参数default，lamdba表达式  
 \* Returns an element at the given* ***[index]*** *or the result of calling the* ***[defaultValue]*** *\* function if the* ***[index]*** *is out of bounds of this collection.  
 \* 2  
 \*/***val elementAtOrElse** = **list**.*elementAtOrElse*(2) **{ "error" }***/\*\*  
 \* 返回给定index对应的元素，如果index数组越界则会 返回null  
 \* Returns an element at the given* ***[index]*** *or `null` if the* ***[index]*** *is out of bounds of this list.  
 \* null  
 \*/***val elementAtOrNull** = **list**.*elementAtOrNull*(19)  
  
*/\*\*  
 \* Returns first element.  
 \** ***@throws [NoSuchElementException]*** *if the list is empty.  
 \* 0  
 \*/***val first** = **list**.*first*()  
  
*/\*\*  
 \* 返回符合给定函数条件的第一个元素,没有回抛异常  
 \* Returns the first element matching the given* ***[predicate]****.  
 \** ***@throws [NoSuchElementException]*** *if no such element is found.  
 \* 4  
 \*/***val first2** = **list**.*first* **{ it** > 3 **}***/\*\*  
 \* 返回符合给定函数条件的第一个元素，如果没有符合则返回null  
 \* Returns the first element matching the given* ***[predicate]****, or `null` if element was not found.  
 \* null  
 \*/***val firstOrNull** = **list**.*firstOrNull* **{ it** > 9 **}***/\*\*  
 \* 返回第一个符合给定函数条件的元素的index，如果没有符合则返回-1  
 \* Returns index of the first element matching the given* ***[predicate]****, or -1 if the list does not contain such element.  
 \* 0  
 \*/***val indexOfFirst** = **list**.*indexOfFirst* **{ it** % 3 == 0 **}***/\*\*  
 \* 返回最后一个符合给定函数条件的元素的index，如果没有符合则返回-1  
 \* Returns index of the last element matching the given* ***[predicate]****, or -1 if the list does not contain such element.  
 \* 11  
 \*/***val indexOfLast** = **list**.*indexOfLast* **{ it** % 3 == 0 **}***/\*\*  
 \* 返回符合给定函数条件的最后一个元素,没有抛异常  
 \* Returns the last element matching the given* ***[predicate]****.  
 \** ***@throws [NoSuchElementException]*** *if no such element is found.  
 \* 6  
 \*/***val last** = **list**.*last* **{ it** > 4 **}***/\*\*  
 \* 返回指定元素的最后一个index，如果不存在，则返回-1  
 \* Returns the index of the last occurrence of the specified element in the list, or -1 if the specified  
 \* element is not contained in the list.  
 \* 8  
 \*/***val lastIndexOf** = **list**.lastIndexOf(3)  
  
*/\*\*  
 \* 返回符合给定函数条件的最后一个元素，如果没有符合则返回null  
 \* Returns the last element matching the given* ***[predicate]****, or `null` if no such element was found.  
 \* null  
 \*/***val lastOrNull** = **list**.*lastOrNull* **{ it** > 8 **}***/\*\*  
 \* 返回符合给定函数的单个元素，如果没有符合或者超过一个，则抛出异常  
 \* Returns the single element matching the given* ***[predicate]****, or throws exception if there is no or more than one matching element.  
 \* 6  
 \*/***val single** = **list**.*single* **{ it** > 5 **}***/\*\*  
 \* 返回符合给定函数的单个元素，如果没有符合或者超过一个，则返回null  
 \* Returns the single element matching the given* ***[predicate]****, or `null` if element was not found or more than one element was found.  
 \* null  
 \*/***val singleOrNull** = **list**.*singleOrNull* **{ it** > 8 **}**

## 生产操作符

**private val list1**= *listOf*(0,1,2,3,4,5)  
 **private val list2**= *listOf*(4,5,2,1,5)  
  
 */\*\*  
 \* 返回两个数组对应index位置组成的map的集合  
 \* [(0, 4), (1, 5), (2, 2), (3, 1), (4, 5)]  
 \*/* **val zip**= **list1**.*zip*(**list2**)  
  
 */\*\*  
 \* 返回两个数组对应index的值符合处理函数结果的值组成的数组  
 \* [4, 6, 4, 4, 9]  
 \*/* **val zip2**= **list1**.*zip*(**list2**)**{**it1,it2**->**it1+it2**}***//返回自己琢磨，应该是交集  
 //[(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (5, 5)]* **val zip4**= **list1**.*zip*(**list1**+**list2**)  
 *//数组相加，再返回符合函数的结果数组  
 //[0, 2, 4, 6, 8, 10]* **val zip3**= **list1**.*zip*(**list1**+**list2**)**{**it1,it2**->**it1+it2**}** */\*\*  
 \* Returns a pair of lists, where  
 \* \*first\* list is built from the first values of each pair from this collection,  
 \* \*second\* list is built from the second values of each pair from this collection.  
 \* ([1, 3, 7], [3, 4, 8])  
 \*/* **val unzip** = *listOf*(Pair(1, 3), Pair(3, 4), Pair(7, 8)).*unzip*()  
  
  
 */\*\*  
 \* 返回一个包含原集合和给定集合中所有元素的集合，因为函数的名字原因，我们可以使用+操作符。  
 \* Returns a list containing all elements of the original collection and then all elements of the given* ***[elements]*** *sequence.  
 \* [0, 1, 2, 3, 4, 5, 4, 5, 2, 1, 5]  
 \*/* **val plus** = **list1**.*plus*(**list2**)

## 顺序操作符

**private val list** = *listOf*(1, 8, 9, 4, 5, 6, 4, 5, 3, 8, 5)  
  
*/\*\*  
 \* 返回一个与指定list相反顺序的list  
 \* Returns a list with elements in reversed order.  
 \*/***val reverse** = **list**.*reversed*()  
  
*/\*\*  
 \* 返回一个自然排序后的list  
 \* Returns a list of all elements sorted according to their natural sort order.  
 \*/***val sort** = **list**.*sorted*()  
  
*/\*\*  
 \* 返回一个根据指定函数排序后的list  
 \* Returns a list of all elements sorted according to natural sort order of the value returned by specified* ***[selector]*** *function.  
 \*/***val sortBy** = **list**.*sortedBy* **{ it** > 3 **}***/\*\*  
 \* 返回一个降序排序后的List  
 \* Returns a list of all elements sorted descending according to their natural sort order.  
 \*/***val sortDescending** = **list**.*sortedDescending*()  
  
*/\*\*  
 \* 返回一个根据指定函数降序排序后的list  
 \* Returns a list of all elements sorted descending according to natural sort order of the value returned by specified* ***[selector]*** *function.  
 \*/***val sortDescendingBy** = **list**.*sortedByDescending* **{ it** > 4 **}**