**正则表达式之捕获组和非捕获组**

2018年01月23日 15:08:12

阅读数：91

关于捕获的一些主要用法

1. (?:X)
2. (?=X)
3. (?<=X)
4. (?!X)
5. (?<!X)

捕获

要书明白捕获，就要先从分组开始。重复单字符我们可以使用限定符，如果重复字符串，用什么呢？ 对！用小括号，小括号里包裹指定字表达式（子串），这就是分组。之后就可以限定这个子表示的重复次数了。

那么，什么是捕获呢？使用小括号指定一个子表达式后，匹配这个子表达式的文本（即匹配的内容）可以在表达式或者其他过程中接着用，怎么用呢？至少应该有个指针啥的引用它吧？ 对！默认情况下，每个分组（小括号）会自动

拥有一个组号,从左到右，以分组的左括号为标志，第一个出现的分组组号为1，后续递增。如果出现嵌套，

例如：

“aabcd”

采用正则 (a(b))(c) match 结果入下：

|  |  |
| --- | --- |
| **分组** | **捕获** |
| $1(group1) | ab |
| $2(group2) | b |
| $3(group3) | c |

继续涨姿势。

一、(?:)非捕获组走起。

由下面一个例子引发对非捕获组的学习。

有两个金额：6000￥ 和 1000$。

需求是得到金额和货币种类。

『废话少说，多用正则』：

(\d+)+([$￥])

输出结果为：

6000  $

OK,满足了要求。这里的正则分成了两个组，一个是(d+)，一个是(￥$)，前一个组($1)匹配金额，后一个组($2)匹配货币种类。

现在，需求变了!! 我需要这个正则同时可以匹配浮点数小数点前面的整数。如10010.86￥，提炼出 10010 和 ￥。

那么正则如下：

(\d+)(\.?)(\d)([$￥])

这里用括号分了四组，所以要输出金额的整数部分和种类，要分别输了$1,$4了。如果输出部分和正则是分开的，我希望只修改正则而不去修改输出部分的代码，也就是还是用$1，$2作为输出。由此可以引出非捕获组(?:)。

把前面的正则修改为：

(\d+)(?:\.?)(?:\d+)([￥$])$

这样，还是用$1,$2做为输出，同样输出了 10010 和 ￥

这个正则的中间两个组用到的就是非捕获组（?:），它可以理解为只分组而不捕获。

二、(?=)和(?<=) 前后查找

有的资料把它们叫做肯定式向前查找和肯定式向后查找；

有的资料也叫做肯定顺序环视和肯定逆序环视。

1、直接看下面的例子：

[0-9a-z]{2}(?=aa) var str = "12332aa438aaf";

Match List:

|  |  |
| --- | --- |
| 1 | 32 |
| 2 | 38 |

这个正则的意思是：匹配这么一个字符串，它要满足：是两位字符（数字，或字母），且后面紧跟着两个a。

分析一下：

32aa 这个子串满足这个条件，所以可以匹配到，又因为 (?=) 的部分是不捕获的，所以输出的只是 32，不包括aa。同理 38aa 也匹配这个正则，而输出仅是 38。

再深入看一下:

当str第一次匹配成功输出 32 后，程序要继续向后查找是否还有匹配的其它子串。那么这时应该从 32aa 的后一位开始向后查找，还是从 32 的后一位呢？也就是从索引 5 开始还是从 7 开始呢？有人可能想到是从 32aa 的下一位开始往后找，因为 32aa 匹配了正则，所以下一位当然是它的后面也就是从 4 开始。但实际上是从 32 的后一位也就是第一个 a 开始往后找。原因还是 (?=) 是非捕获的。

查阅API文档是这么注释的：

(?=X) X, via zero-width positive lookahead

可见zero-width（零宽度）说的就是这个意思。

现在，把字符串写的更有意思些：str = "aaaaaaaa";

看一下它的输出： aa aa aa

分析一下：

这个字符串一共有8个a。

第一次匹配比较容易找到，那就是前四个：aaaa ,当然第三和第四个 a 是不捕获的，所以输出是第一和第二个a；

接着继续查找，这时是从第三个a开始，三到六，这4个a区配到了，所以输出第三和第四个a；

接着继续查找，这时是从第五个a开始，五到八，这4个a区配到了，所以输出第五和第六个a；

接着往后查找，这时是从第七个a开始，显然，第七和第八个a,不满足正则的匹配条件，查找结束。

我们再延伸一下，刚说的情况的是(?=)放在捕获的字符串后面，它如果放在前面又是什么结果呢？

例子换成：

(?=hopeful)hope

它的输出是hope。

正则的意思是：是否能匹配hopeful,如果能，则捕获hopeful中的hope。当然继续向后查找匹配的子串，是从f开始。

比较一下可以看出，(?=hopeful)hope 和 hope(?=ful),两个正则的效果其实是一样的。

2、下面说一下 (?<=)

把正则改一下，

(?<=aa)[0-9a-z]{2};

字符串还是str = "12332aa438aaf";

它的输出：43。

这个正则的意思是：匹配这么一个字符串，它要满足：是两位字符（数字或字母），且前面紧跟的是两个字母 a。

同样，深入一下，把str换成str = "aaaaaaaa";看一下输出是什么，同样也是：aa aa aa

分析一下：

第一次匹配不用说，是前四个a，输出的是第三和第四个a;

继续向后查找，从第五个a开始，程序发现，第五个和第六个a满足，因为是两位字符，且满足前面紧跟着两个a(第三和第四个a)。所以匹配成功，输出第五个和第六个a;

继续向后查找，从第七个a开始，程序发现，第七个和第八个a满足，因为是两位字符，且满足前面紧跟着两个a(第五和第六个a)。所以匹配成功，输出第七和第八个a。查找结束。

三、(?!)和(?<!) 逆袭!

从外观上看，和前面一组很相似，区别就是把 ‘=’ 换成了 ‘!’

那么意义刚好也是相反的。

[0-9a-z]{2}(?!aa)意思是：匹配两个字符，且后面紧跟着的不是aa

(?<!aa)[0-9a-z]{2} 意思是：匹配两个字符，且前面紧跟着的不是aa