正则表达式简明参考

thinkphp

前言

> 来源：http://www.xiaoleilu.com/regex-guide/
> 作者：路小磊

我想在网上最出名的正则相关的一篇文章就是《正则表达式30分钟入门教程》了，说实话这篇文章确实是我的正则入门，但是随着使用熟练，冗长的文章已经不能满足我了，在此做个归纳总结，用于快速查阅。

以下语法在Java中有效，大部分应该是通用的。

元字符

元字符，又叫字符集，就是用一些特殊符号表示特定种类的字符或位置。

匹配字符

- `.` 匹配除换行符以外的任意字符
- `\w` 匹配字母或数字或下划线或汉字
- `\s` 匹配任意的空白符
- `\d` 匹配数字

匹配位置

- `\b` 匹配单词的开始或结束
- `^` 匹配字符串的开始
- `$` 匹配字符串的结束
- `\G` 上一个匹配的结尾（本次匹配开始）
- `\A` 字符串开头(类似 `^`，但不受处理多行选项的影响)
- `\Z` 字符串结尾或行尾(不受处理多行选项的影响)
- `\z` 字符串结尾(类似 `$`，但不受处理多行选项的影响)

重复

- `*` 重复零次或更多次
- `+` 重复一次或更多次
- `?` 重复零次或一次
- `{n}` 重复n次
- `{n,}` 重复n次或更多次
- `{n,m}` 重复n到m次

字符转义

如果想匹配元字符本身或者正则中的一些特殊字符，使用 `\` 转义。例如匹配 `*` 这个字符则使用 `\*`，匹配 `\` 这个字符，使用 `\\`。

需要转义的字符：`$`, `(`, `)`, `*`, `+`, `.`, `[`, `]`, `?`, `\`, `^`, `{`, `}`, `|`

字符类

当需要匹配明确的字符或字符集合时候，就用到字符类。

特殊字符

- `\0hh` 8进制值hh所表示的字符
- `\xhh` 16进制值hh所表示的字符
- `\uhhhh` 16进制值hhhh所表示的Unicode字符
- `\t` Tab
- `\n` 换行符
- `\r` 回车符
- `\f` 换页符
- `\e` Escape
- `\cN` ASCII控制字符。比如`\cC`代表`Ctrl+C`
- `\p{name}` Unicode中命名为name的字符类，例如`\p{IsGreek}`

陈列

- `[aeiou]` 匹配一个元音字符
- `[.?!]` 匹配给定的一个标点

范围

- `[0-9]` 匹配0~9的数字，同`\d`
- `[a-z]` 匹配所有小写字母
- `[a-zA-Z]` 匹配所有字母
- `[a-z0-9A-Z_]` 等同于`\w`

反义

表示不属于元字符或者字符类的字符

反义元字符

- `\W` 匹配任意不是字母，数字，下划线，汉字的字符
- `\S` 匹配任意不是空白符的字符
- `\D` 匹配任意非数字的字符
- `\B` 匹配不是单词开头或结束的位置

反义字符类

- `[^x]` 匹配除了x以外的任意字符
- `[^aeiou]` 匹配除了aeiou这几个字母以外的任意字符

分支条件

又叫逻辑运算符，在此 X 和 Y 表示两个**表达式**

- XY X紧跟Y
- X|Y 表示X或Y，从左到右，满足第一个条件就不会继续匹配了。

分组

在这里我把**表达式**统一以 \w 为例：

- (\w) 被一个括号包围起来是一个整体，表示一个分组
- (\w)(\w) 自动命名分组，第一个小括号是分组1，第二个小括号是分组2
- (?'Word'\w+)) 表示定义了一个叫做 Word 的分组
- (?\w+)) 表示定义了一个叫做 Word 的分组
- (?:\w+) 匹配exp,不捕获匹配的文本，也不给此分组分配组号

反向引用

后面的表达式可以引用前面的某个分组，用 \1 表示，就好像分组1的值赋值给了 \1 这个变量，这个变量可以在后面任意位置引用。

- \1 表示分组1匹配的文本
- \k 表示分组 Word 匹配的文本

匹配重复两个的英文，例如匹配 Hello Hello 、 lei123 lei123 ：

1. (\w+)\s+\1
2. (?\w+)\s+\k

零宽断言（正向和负向）

零宽断言表示匹配字符的时候再添加一些定位条件，使匹配更精准。

- \w+(?=ing) 匹配以 ing 结尾的多个字符（不包括ing）
- \w+(?!ing) 匹配**不是**以 ing 结尾的多个字符
-

- (?<=re)\w+ 匹配以 re 开头的多个字符（不包括re）
- (?<!re)\w+ 匹配**不是**以 re 开头的多个字符
-

- (?<=\s)\d+(?=\s) 匹配两边是空白符的数字，不包括空白符

贪婪与懒惰

贪婪：匹配尽可能长的字符串

懒惰：匹配尽可能短的字符串

懒惰模式的启用只需在**重复元字符**之后加`?`既可。

- `*?` 重复任意次，但尽可能少重复
- `+?` 重复1次或更多次，但尽可能少重复
- `??` 重复0次或1次，但尽可能少重复
- `{n,m}?` 重复n到m次，但尽可能少重复
- `{n,}?` 重复n次以上，但尽可能少重复

处理选项

在表达式里插记号的方式来启用绝大多数的模式，在正则的哪里插入，就从哪里启用。

1. `(?i)`：忽略大小写(CASE_INSENSITIVE)
2. `(?x)`：忽略空格字符(COMMENTS)
3. `(?s)`：`.`匹配任意字符，包括换行符（DOTALL）
4. `(?m)`：多行模式（MULTILINE）
5. `(?u)`：对Unicode符大小写不敏感（UNICODE_CASE），必须启用**CASE_INSENSITIVE**
6. `(?d)`：只有'\n'才被认作一行的中止（UNIX_LINES）

平衡组/递归匹配

平衡组用于匹配嵌套层次结构，常用于匹配HTML标签（当HTML内容不规范，起始标签和结束标签数量不同时，匹配出正确配对的标签），在此把表达式统一以`\w`为例。

- `(?'group'\w)` 捕获的分组（`\w`匹配到的内容）命名为`group`，并压入堆栈
- `(?'-group'\w)` 捕获分组（`\w`匹配到的内容）后，弹出`group`分组栈的栈顶内容（最后压入的捕获内容），堆栈本来为空，则本分组的匹配失败
- `(?(group)yes|no)` 如果`group`栈非空匹配表达式`yes`，否则匹配表达式`no`
- `(?!)` 零宽负向先行断言，由于没有后缀表达式，试图匹配总是失败

注释

注释语法：`(?#comment)`，这个语法的内容会被正则忽略，用于注释含义。可以放在正则表达式的任意位置。

参考

正则表达式30分钟入门教程

正则表达式