

正则表达式

原作者github: <https://github.com/CyC2018/CS-Notes>

PDF制作github: <https://github.com/sjsdfg/CS-Notes-PDF>

一、概述

正则表达式用于文本内容的查找和替换。

正则表达式内置于其它语言或者软件产品中，它本身不是一种语言或者软件。

[正则表达式在线工具](#)

二、匹配单个字符

. 可以用来匹配任何的单个字符，但是在绝大多数实现里面，不能匹配换行符；

. 是元字符，表示它有特殊的含义，而不是字符本身的含义。如果需要匹配 . ，那么要用 \ 进行转义，即在 . 前面加上 \ 。

正则表达式一般是区分大小写的，但是也有些实现是不区分。

正则表达式

```
nam.
```

匹配结果

My name is Zheng.

三、匹配一组字符

[] 定义一个字符集合；

0-9、a-z 定义了一个字符区间，区间使用 ASCII 码来确定，字符区间在 [] 中使用。

- 只有在 [] 之间才是元字符，在 [] 之外就是一个普通字符；

^ 在 [] 中是取非操作。

应用

匹配以 abc 为开头，并且最后一个字母不为数字的字符串：

正则表达式

```
abc[^0-9]
```

匹配结果

1. abcd

2. abc1
3. abc2

四、使用元字符

匹配空白字符

元字符	说明
<code>[\b]</code>	回退（删除）一个字符
<code>\f</code>	换页符
<code>\n</code>	换行符
<code>\r</code>	回车符
<code>\t</code>	制表符
<code>\v</code>	垂直制表符

`\r\n` 是 Windows 中的文本行结束标签，在 Unix/Linux 则是 `\n`。

`\r\n\r\n` 可以匹配 Windows 下的空白行，因为它将匹配两个连续的行尾标签，而这正是两条记录之间的空白行；

匹配特定的字符类别

1. 数字元字符

元字符	说明
<code>\d</code>	数字字符，等价于 <code>[0-9]</code>
<code>\D</code>	非数字字符，等价于 <code>[^0-9]</code>

2. 字母数字元字符

元字符	说明
<code>\w</code>	大小写字母，下划线和数字，等价于 <code>[a-zA-Z0-9_]</code>
<code>\W</code>	对 <code>\w</code> 取非

3. 空白字符元字符

元字符	说明
<code>\s</code>	任何一个空白字符，等价于 <code>[\f\n\r\t\v]</code>
<code>\S</code>	对 <code>\s</code> 取非

`\x` 匹配十六进制字符，`\o` 匹配八进制，例如 `\x0A` 对应 ASCII 字符 10，等价于 `\n`。

五、重复匹配

- `+` 匹配 1 个或者多个字符
- `*` 匹配 0 个或者多个
- `?` 匹配 0 个或者 1 个

应用

匹配邮箱地址。

正则表达式

```
[\\w.]+@\\w+\\.\\w+
```

`[\\w.]` 匹配的是字母数字或者 `.`，在其后面加上 `+`，表示匹配多次。在字符集合 `[]` 里，`.` 不是元字符；

匹配结果

abc.def@qq.com

- `{n}` 匹配 `n` 个字符
- `{m, n}` 匹配 `m~n` 个字符
- `{m,}` 至少匹配 `m` 个字符

`*` 和 `+` 都是贪婪型元字符，会匹配最多的内容。在后面加 `?` 可以转换为懒惰型元字符，例如 `*?`、`+?` 和 `{m, n}?`。

正则表达式

```
a.+c
```

由于 `+` 是贪婪型的，因此 `.+` 会匹配更多可能的内容，所以会把整个 `abcabcabc` 文本都匹配，而不是只匹配前面的 `abc` 文本。用懒惰型可以实现匹配前面的。

匹配结果

abcabcabc

六、位置匹配

单词边界

`\b` 可以匹配一个单词的边界，边界是指位于 `\w` 和 `\W` 之间的位置；`\B` 匹配一个不是单词边界的位置。

`\b` 只匹配位置，不匹配字符，因此 `\babc\b` 匹配出来的结果为 3 个字符。

字符串边界

^ 匹配整个字符串的开头，\$ 匹配结尾。

^ 元字符在字符集中用作求非，在字符集外用作匹配字符串的开头。

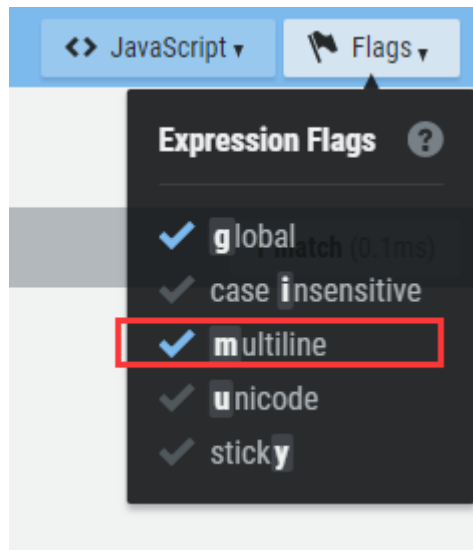
分行匹配模式 (multiline) 下，换行被当做字符串的边界。

应用

匹配代码中以 // 开始的注释行

正则表达式

```
^\s*\//\.*$
```



匹配结果

```
1. public void fun() {  
2.   // 注释 1  
3.   int a = 1;  
4.   int b = 2;  
5.   // 注释 2  
6.   int c = a + b;  
7. }
```

七、使用子表达式

使用 () 定义一个子表达式。子表达式的内容可以当成一个独立元素，即可以将它看成一个字符，并且使用 * 等元字符。

子表达式可以嵌套，但是嵌套层次过深会变得很难理解。

正则表达式

```
(ab){2,}
```

匹配结果

ababab

| 是或元字符，它把左边和右边所有的部分都看成单独的两个部分，两个部分只要有一个匹配就行。

正则表达式

```
(19|20)\d{2}
```

匹配结果

1. 1900
2. 2010
3. 1020

应用

匹配 IP 地址。

IP 地址中每部分都是 0-255 的数字，用正则表达式匹配时以下情况是合法的：

- 一位数字
- 不以 0 开头的两位数字
- 1 开头的三位数
- 2 开头，第 2 位是 0-4 的三位数
- 25 开头，第 3 位是 0-5 的三位数

正则表达式

```
((25[0-5]|(2[0-4]\d)|(1\d{2})|([1-9]\d)|(\d))\.)\d{3}((25[0-5]|(2[0-4]\d)|(1\d{2})|([1-9]\d)|(\d)))
```

匹配结果

1. 192.168.0.1
2. 00.00.00.00
3. 555.555.555.555

八、回溯引用

回溯引用使用 \n 来引用某个子表达式，其中 n 代表的是子表达式的序号，从 1 开始。它和子表达式匹配的内容一致，比如子表达式匹配到 abc，那么回溯引用部分也需要匹配 abc。

应用

匹配 HTML 中合法的标题元素。

正则表达式

\1 将回溯引用子表达式 (h[1-6]) 匹配的内容，也就是说必须和子表达式匹配的内容一致。

```
<(h[1-6])>\w*?<\/\1>
```

匹配结果

1. <h1>x</h1>
2. <h2>x</h2>
3. <h3>x</h1>

替换

需要用到两个正则表达式。

应用

修改电话号码格式。

文本

313-555-1234

查找正则表达式

```
(\d{3})(-)(\d{3})(-)(\d{4})
```

替换正则表达式

在第一个子表达式查找的结果加上 ()，然后加一个空格，在第三个和第五个子表达式查找的结果中间加上 - 进行分隔。

```
(\d1) \d3-\d5
```

结果

(313) 555-1234

大小写转换

元字符	说明
\l	把下个字符转换为小写
\u	把下个字符转换为大写
\L	把\L和\E之间的字符全部转换为小写
\U	把\U和\E之间的字符全部转换为大写
\E	结束\L或者\U

应用

把文本的第二个和第三个字符转换为大写。

文本

abcd

查找

```
(\w)(\w{2})(\w)
```

替换

```
$1\U$2\E$3
```

结果

aBCd

九、前后查找

前后查找规定了匹配的内容首尾应该匹配的内容，但是又不包含首尾匹配的内容。向前查找用 `?=` 来定义，它规定了尾部匹配的内容，这个匹配的内容在 `?=` 之后定义。所谓向前查找，就是规定了一个匹配的内容，然后以这个内容为尾部向前面查找需要匹配的内容。向后匹配用 `?<=` 定义（注：JavaScript 不支持向后匹配，Java 对其支持也不完善）。

应用

查找出邮件地址 @ 字符前面的部分。

正则表达式

```
\w+(?=)
```

结果

abc@qq.com

对向前和向后查找取非，只要把 `=` 替换成 `!` 即可，比如 `(?=)` 替换成 `(?!)`。取非操作使得匹配那些首尾不符合要求的内容。

十、嵌入条件

回溯引用条件

条件判断为某个子表达式是否匹配，如果匹配则需要继续匹配条件表达式后面的内容。

正则表达式

子表达式 `\()` 匹配一个左括号，其后的 `?` 表示匹配 0 个或者 1 个。`?(1)` 为条件，当子表达式 1 匹配时条件成立，需要执行 `)` 匹配，也就是匹配右括号。

```
(\())?abc(?(1)\))
```

结果

1. (abc)
2. abc
3. (abc

前后查找条件

条件为定义的首尾是否匹配，如果匹配，则继续执行后面的匹配。注意，首尾不包含在匹配的内容中。

正则表达式

?(?=) 为前向查找条件，只有在以 - 为前向查找的结尾能匹配 \d{5}，才继续匹配 -\d{4}。

```
\d{5}(?(?=-)\d{4})
```

结果

1. 11111
2. 22222-
3. 33333-4444

参考资料

- BenForta. 正则表达式必知必会 [M]. 人民邮电出版社, 2007.

github: <https://github.com/sjsdfg/CS-Notes-PDF>