

04 | 匹配模式：一次性掌握正则中常见的4种匹配模式

2020-06-19 涂伟忠

正则表达式入门课

[进入课程 >](#)



讲述：涂伟忠

时长 13:00 大小 11.91M



你好，我是涂伟忠。今天我们一起学习正则中的匹配模式（Match Mode）。

所谓匹配模式，指的是正则中一些**改变元字符匹配行为**的方式，比如匹配时不区分英文字母大小写。常见的匹配模式有 4 种，分别是不区分大小写模式、点号通配模式、多行模式和注释模式。我们今天主要来讲一下这 4 种模式。

需要注意的是，这里的“模式”对应的是英文中的 mode，而不是 pattern。有些地方会把正则表达式 pattern 也翻译成模式，你在网上看到的技术文章中讲的正则模式，有可能^{其实是}的是正则表达式本身，这一点你需要注意区别。



不区分大小写模式（Case-Insensitive）

首先，我们来看一下不区分大小写模式。它有什么用呢？学一个知识的时候，我一般喜欢先从它的应用出发，这样有时候更能激发我学习的兴趣，也更容易看到学习成果。

下面我来举个例子说明一下。在进行文本匹配时，我们要关心单词本身的意义。比如要查找单词 cat，我们并不需要关心单词是 CAT、Cat，还是 cat。根据之前我们学到的知识，你可能会把正则写成这样：**[Cc][Aa][Tt]**，这样写虽然可以达到目的，但不够直观，如果单词比较长，写起来容易出错，阅读起来也比较困难。

REGULAR EXPRESSION3 matches, 15 steps (~0ms)

⋮ / **[Cc][Aa][Tt]** / gm

TEST STRINGSWITCH TO UNIT TESTS ▶

cat
CAT
Cat

那么有没有更好的办法来实现这个需求呢？这时候不区分大小写模式就派上用场了。

我们前面说了，不区分大小写是匹配模式的一种。当我们把**模式修饰符**放在整个正则前面时，就表示整个正则表达式都是不区分大小写的。模式修饰符是通过 **(? 模式标识)** 的方式来表示的。我们只需要把模式修饰符放在对应的正则前，就可以使用指定的模式了。在不区分大小写模式中，由于不分大小写的英文是 Case-Insensitive，那么对应的模式标识就是 I 的小写字母 i，所以不区分大小写的 cat 就可以写成 **(?i)cat**。

REGULAR EXPRESSION

3 matches, 15 steps (~0ms)

:/ (?i)cat

/ gm

TEST STRING

SWITCH TO UNIT TESTS ▶

cat

CAT

Cat

你看，和[**Cc**][**Aa**][**Tt**] 相比，这样是不是清晰简洁了很多呢？

我们也可以用它来尝试匹配两个连续出现的 cat，如下图所示，你会发现，即便是第一个 cat 和第二个 cat 大小写不一致，也可以匹配上。

REGULAR EXPRESSION v1 ▼

5 matches, 45 steps (~0ms)

:/ (?i)(cat) \1

/ gm

TEST STRING

SWITCH TO UNIT TESTS ▶

cat cat

CAT Cat

Cat cat

Cat Cat

CAT CAT

我给了你一个测试链接，你可以在这里试试不区分大小写模式：

🔗 <https://regex101.com/r/x1lg4P/1>。

如果我们想要前面匹配上的结果，和第二次重复时的大小写一致，那该怎么做呢？我们只需要用括号把**修饰符和正则 cat 部分**括起来，加括号相当于作用范围的限定，让不区分大小写只作用于这个括号里的内容。同样的，我在🔗[这里](#)给你放了一个测试链接，你可以自己看一下。

REGULAR EXPRESSION v2 ▾

3 matches, 57 steps (~0ms)

⋮ /

`((?i)cat) \1`

/ gm 🚩

TEST STRING

SWITCH TO UNIT TESTS ▸

| | |
|-----|-----|
| cat | cat |
| CAT | Cat |
| Cat | cat |
| Cat | Cat |
| CAT | CAT |

需要注意的是，这里正则写成了 `((?i)cat) \1`，而不是 `((?i)(cat)) \1`。也就是说，我们给修饰符和 cat 整体加了个括号，而原来 cat 部分的括号去掉了。如果 cat 保留原来的括号，即 `((?i)(cat)) \1`，这样正则中就会有子组，虽然结果也是对的，但这其实没必要。在上一讲里我们已经讲解了相关的内容，如果忘记了你可以回去复习一下。

到这里，我们再进阶一下。如果用正则匹配，实现部分区分大小写，另一部分不区分大小写，这该如何操作呢？比如说我现在想要，the cat 中的 the 不区分大小写，cat 区分大小写。

通过上面的学习，你应该能很快写出相应的正则，也就是 `((?i)the) cat`。实现的效果如下：

```
:/((?i)the) cat
```

```
/gm
```


TEST STRING

SWITCH TO UNIT TESTS ▶

```
the cat
The cat
THE cat
the Cat
the CAT
```

我把部分区分大小写，部分不区分大小写的测试链接放在 [这里](#)，你可以看一下。

有一点需要你注意一下，上面讲到的通过**修饰符指定匹配模式**的方式，在大部分编程语言中都是可以直接使用的，但在 JS 中我们需要使用 `/regex/i` 来指定匹配模式。在编程语言中通常会提供一些预定义的常量，来进行匹配模式的指定。比如 Python 中可以使用 `re.IGNORECASE` 或 `re.I`，来传入正则函数中表示不区分大小写。我下面给出了你一个示例，你可以看一下。

 复制代码

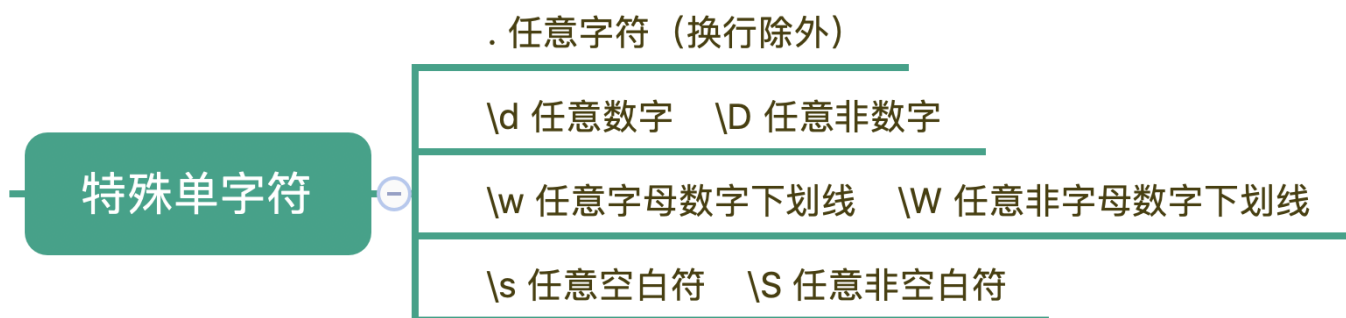
```
1 >>> import re
2 >>> re.findall(r"cat", "CAT Cat cat", re.IGNORECASE)
3 ['CAT', 'Cat', 'cat']
```

到这里我简单总结一下不区分大小写模式的要点：

1. 不区分大小写模式的指定方式，使用模式修饰符 `(?i)`；
2. 修饰符如果在括号内，作用范围是这个括号内的正则，而不是整个正则；
3. 使用编程语言时可以使用预定义好的常量来指定匹配模式。

点号通配模式 (Dot All)

在基础篇的第一讲里，我为你讲解了元字符相关的知识，你还记得英文的点 (.) 有什么用吗？它可以匹配上任何符号，但不能匹配换行。当我们需要匹配真正的“任意”符号的时候，可以使用 `[\s\S]` 或 `[\d\D]` 或 `[\w\W]` 等。



但是这么写不够简洁自然，所以正则中提供了一种模式，让英文的点 (.) 可以匹配上包括换行的任何字符。

这个模式就是**点号通配模式**，有很多地方把它称作单行匹配模式，但这么说容易造成误解，毕竟它与多行匹配模式没有联系，因此在课程中我们统一用更容易理解的“点号通配模式”。

单行的英文表示是 **Single Line**，单行模式对应的修饰符是 **(?s)**，我还是选择用 the cat 来给你举一个点号通配模式的例子。如下图所示：

:/ (?s) .+

点可以匹配上换行



TEST STRING

SWITCH TO UNIT TESTS ▸

the cat

The cat

THE cat

the Cat

the CAT

需要注意的是，JavaScript 不支持此模式，那么我们就可以使用前面说的`[\s\S]`等方式替代。在 Ruby 中则是用 `Multiline`，来表示点号通配模式（单行匹配模式），我猜测设计者的意图是把点（`.`）号理解成“能匹配多行”。

多行匹配模式 (Multiline)

讲完了点号通配模式，我们再来看看多行匹配模式。通常情况下，`^` 匹配整个字符串的开头，匹配整个字符串的结尾。多行匹配模式改变的就是^和的匹配行为。

REGULAR EXPRESSION v2 ▾

2 matches, 62 steps (~0ms)

:/ ^the|cat\$/g

TEST STRING

SWITCH TO UNIT TESTS ▸

the little cat
the small cat

^ 匹配整个文本的开头
\$ 匹配整个文本的结尾

非多行模式下

多行模式的作用在于，使 ^ 和 \$ 能匹配上**每行**的开头或结尾，我们可以使用模式修饰符号 (?m) 来指定这个模式。

REGULAR EXPRESSION v2 ▾

4 matches, 80 steps (~0ms)

:/ (?m)^the|cat\$/g

TEST STRING

SWITCH TO UNIT TESTS ▸

the little cat
the small cat

^ 匹配了每行的开头
\$ 匹配了每行的结尾

多行匹配模式下

这个模式有什么用呢？在处理日志时，如果日志以时间开头，有一些日志打印了堆栈信息，占用了多行，我们就可以使用多行匹配模式，在日志中匹配到以时间开头的每一行日志。

值得一提的是，正则中还有 `\A` 和 `\z` (Python 中是 `\Z`) 这两个元字符容易混淆，`\A` 仅匹配整个字符串的开始，`\z` 仅匹配整个字符串的结束，在多行匹配模式下，它们的匹配行为不会改变，如果只想匹配整个字符串，而不是匹配每一行，用这个更严谨一些。

注释模式 (Comment)

在实际工作中，正则可能会很复杂，这就导致编写、阅读和维护正则都会很困难。我们在写代码的时候，通常会在一些关键的地方加上注释，让代码更易于理解。很多语言也支持在正则中添加注释，让正则更容易阅读和维护，这就是正则的注释模式。正则中注释模式是使用 `(?#comment)` 来表示。

比如我们可以把单词重复出现一次的正则 `(\w+)\1` 写成下面这样，这样的话，就算不是很懂正则的人也可以通过注释看懂正则的意思。

```
1 (\w+)(?#word) \1(?#word repeat again)
```

复制代码

REGULAR EXPRESSION v2

3 matches, 84 steps (~0ms)

/ **(\w+)**(?#word) **\1**(?#word repeat again) / g

TEST STRING SWITCH TO UNIT TESTS

cat cat
CAT Cat
Cat cat
Cat Cat
CAT CAT

在很多编程语言中也提供了 `x` 模式来书写正则，也可以起到注释的作用。我用 Python3 给你举了一个例子，你可以参考一下。

```
1 import re
2
3 regex = r'''(?mx) # 使用多行模式和x模式
4 ^           # 开头
5 (\d{4})     # 年
6 (\d{2})     # 月
7 $          # 结尾
8 '''
9
10 re.findall(regex, '202006\n202007')
11 # 输出结果 [('2020', '06'), ('2020', '07')]
```

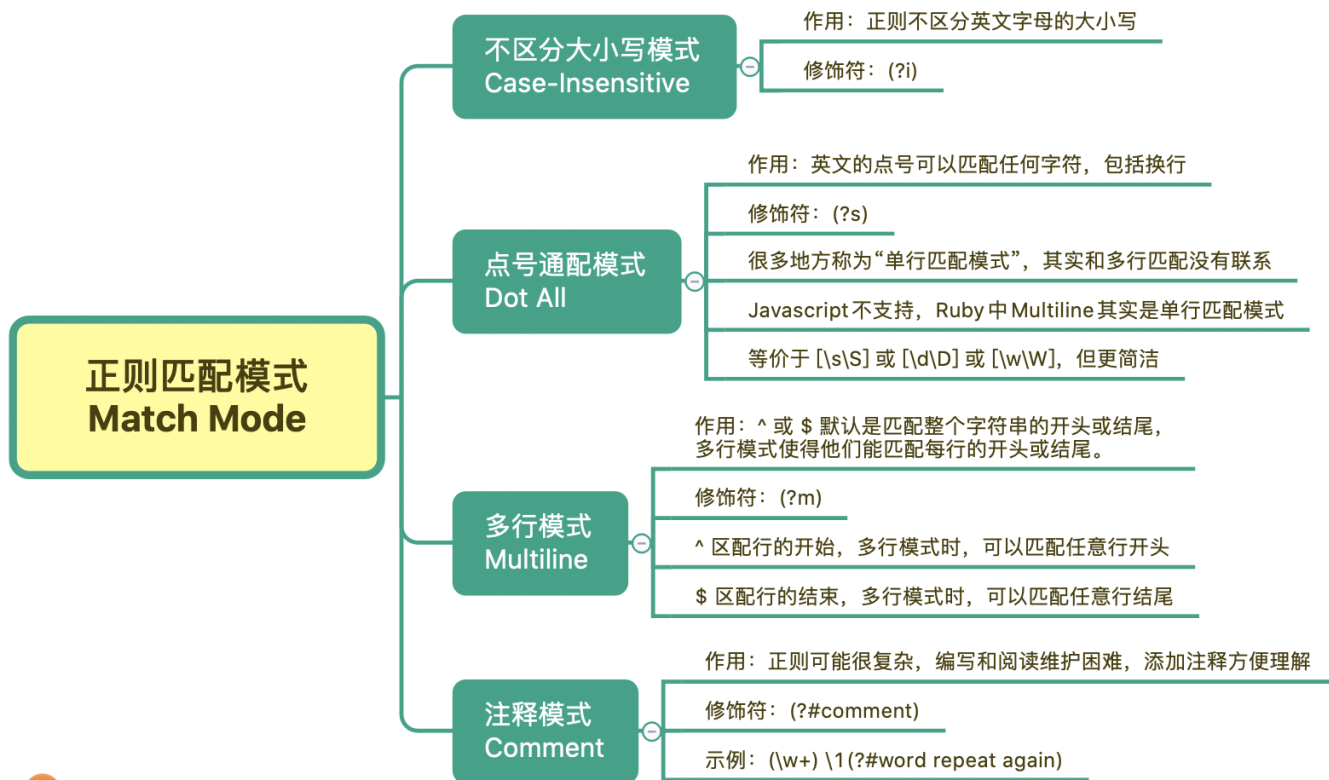
需要注意的是在 x 模式下，所有的换行和空格都会被忽略。为了换行和空格的正确使用，我们可以通过把空格放入字符组中，或将空格转义来解决换行和空格的忽略问题。我下面给了你一个示例，你可以看看。

```
1 regex = r'''(?mx)
2 ^           # 开头
3 (\d{4})     # 年
4 [ ]        # 空格
5 (\d{2})     # 月
6 $          # 结尾
7 '''
8
9 re.findall(regex, '2020 06\n2020 07')
10 # 输出结果 [('2020', '06'), ('2020', '07')]
```

总结

最后，我来给你总结一下，正则中常见的四种匹配模式，分别是：不区分大小写、点号通配模式、多行模式和注释模式。

1. 不区分大小写模式，它可以让整个正则或正则中某一部分进行不区分大小写的匹配。
2. 点号通配模式也叫单行匹配，改变的是点号的匹配行为，让其可以匹配任何字符，包括换行。
3. 多行匹配说的是 ^ 和 \$ 的匹配行为，让其可以匹配上每行的开头或结尾。
4. 注释模式则可以在正则中添加注释，让正则变得更容易阅读和维护。



思考题

最后，我们来做一个小练习吧。HTML 标签是不区分大小写的，比如我们要提取网页中的 head 标签中的内容，用正则如何实现呢？

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>学习正则</title>
5  </head>
6  <body>
7
8  </body>
9  </html>
```

你可以动手试一试，用文本编辑器或你熟悉的编程语言来实现，经过不断练习你才能更好地掌握学习的内容。

今天的课程就结束了，希望可以帮助到你，也希望你在下方的留言区和我参与讨论，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 03 | 分组与引用：如何用正则实现更复杂的查找和替换操作？

下一篇 05 | 断言：如何用断言更好地实现替换重复出现的单词？

精选留言 (32)

写留言



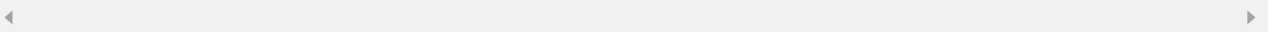
furuhata

2020-06-19

`(?si)<head>(.*<\head>`

展开 ∨

作者回复: 对的，重要是点要能匹配换行，head不区分大小写



3



William

2020-06-23

刊误

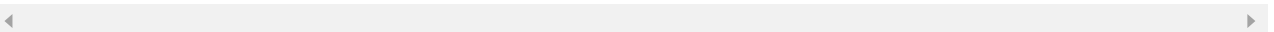
JavaScript已经支持单行模式了。支持 gimsuy 共6个flag。

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/flags

...

展开 ∨

作者回复: 好的，感谢指出，我是在regex101上测试的，发现报错 pattern error，没有细看。看了您提供的文档，发现 ES2018 新加了这个功能。



1



Harmony Chang

2020-06-20

`(?si)<head>.+</head>`

顺便问一下，怎么把<head>和</head>这2个过滤掉，就是只取他们直接的内容。

作者回复: 这个可以后面的在断言里面会讲，下周一就能看到了

1

1



KaKa

2020-06-20

老师能不能多讲解下 实际应用啊 好比我是前端，更主要关注的是js方面的。可是听了几节你将的课以后，还是不知道如何在js中使用。也仅仅只会在你给的一个专门链接里进行测试。可是在js中呢 真的是一脸懵逼。可能是我太菜了[旺柴]，也正是因为我不菜 所以我就买了这门课程 哈哈哈哈哈，麻烦老师回我一下啊

展开

作者回复: 先不要纠结具体的编程语言，要摆脱了字符的限制，深入到概念思维的层面。掌握了里面的各种概念之后，具体怎么写查一查文档，你肯定能表示出来。

文章中给出的链接，在练习的时候，页面左侧有 FLAVOR，ECMAScript (JavaScript)，你可以切换到这个去练习。

2

1



HardToGiveaName

2020-06-19

课后练习题：

`(?is)<head>.*</head>`

在将多行视作单行匹配相关内容，采用单行模式

作者回复: 单行模式的意思是 点匹配所有（包括换行）。不过你说的“将多行视为单行”好像有那么点意思。

正则没问题，可以继续考虑下 head 中有属性的情况。

比如 `<head id="my-head">xxxx</head>`

你这个正则还能继续工作不？



一步

2020-06-19

PCRE: `(?i)<head>(?s).+<\head>`

在 JS 中不支持 `?i` 模式的书写形式: `/<head>.+<\head>/igs`

展开

作者回复: 没问题, 不过这个题目是要求你提取出里面的内容, 不是查找到了就行。另外可以再想一下, 如果 `head` 标签中有属性呢, 比如下面这样, 又该怎么写呢?

`<head id="title">和伟忠一起学习正则表达式</head>`

支持的属性可以看这里 https://www.w3school.com.cn/tags/html_ref_standardattributes.asp



虹炎

2020-06-19

我的答案:

第一步查找: `(?i)<head>((?s).+)<\head>`

第二步替换: `\1`

为什么是`\1` 而不是`\2`, 我觉得是模式修饰符的括号不算分组。老师, 你怎么看?

展开

作者回复: 没问题, 模式修饰符的括号不算分组



虹炎

2020-06-19

老师, 正则默认是多行匹配吗? 我在您提供的链接上测试是。 `^the|cat$` 我不加 `(?m)` 也可以匹配到多行!

作者回复: 看一下后面有没有`gm`之类的, 那个网站可以在后面指定多行模式



前端路上的小学生

2020-06-24

课后习题答案: `(?i)<head>([\s\S]*)</head>`

在JavaScript下: `/<head>([\s\S]*)</head>/gi` 需要使用 `RegExp.$1` 取出值

展开 ▾



前端路上的小学生

2020-06-23

`/(<head>)[\s\S]+</head>/gi`

javascript 语言下使用

展开 ▾



宁悦

2020-06-23

`(?i)^(<head>(?!s).+</head>)$`

展开 ▾



Billions

2020-06-23

使用sublime, 勾选:Case Sensitive. 然后键入: `(?s)<head>.*</head>`

作者回复: 可以的, 找到之后剪切出来, 再把<head>和</head>部分替换去掉, 就相当于提取出来了



小美

2020-06-23

老师, 您多行匹配那一部分文章, 在编辑时被转译成了katex格式了, 还有文章最下面的脑图, 还是关于多行匹配的, 您文字, 匹配, 写成了区配。

展开 ▾

作者回复: 感谢指出, 我找编辑修改一下



Peace



2020-06-23

(?si)<(head).*?>.*?<\1>

展开 ▾



Geek_bbbd6a

2020-06-23

老师请问下，
一个有json语法错误的字符串，如下：

```
"closeBtn":"close",  
},...
```

展开 ▾



Isaac

2020-06-22

```
(?si)<head(\s(profile|accesskey|class|contenteditable|contextmenu|data-[\w\d]|dir|draggable|dropzone|hidden|id|lang|spellcheck|style|tabindex|title|translate)(=".*"?))*>.*<\head>
```

<https://regex101.com/r/x1lg4P/6...>

展开 ▾



seerjk

2020-06-21

v1: <(?:(?i)head)>(?!#case insensitive <head>)((?s).+)(?!#match \n)<\1>(?!#case insensitive </head>)

v1版本比较复杂，从做向右考虑：<(?:(?i)head)> 匹配不区分大小写的 <head> 并不保存分组；((?s).+) 点号通配模式，匹配换行；<\1> 匹配不区分大小写的 <\head> 并不保存分组；...

展开 ▾

作者回复：赞，思路清晰



设置昵称

2020-06-20

js里面我一般用 `/(?<=<head.*?>)(.*?)(?=<\head>)/i`

展开 ▾

作者回复: 赞, 相当于提已经提前把下一节要讲的断言给用上了



chengzise

2020-06-19

课后思考: `(?is)^(<head>\s+(.)\s+<\head>`

参考代码: <https://regex101.com/r/Xavyfv/3>

不同语言的正则细节上还是有一些区别的, 这个需要多练习.

作者回复: 对的, 不同的语言有一些小区别, 不过思路都是一样的, 开始不要纠结在这些细节上, 把主要的功能点都掌握了, 后面要用到的时候, 细节可以理查文档和测试下就好了。

另外可以考虑一下 html 如果是压缩了的, `<head>` 后面没有空白, 你这个正则还能工作么? 比如这样

<https://regex101.com/r/Xavyfv/4>

2



我行我素

2020-06-19

`(?i)<head>(\D*|\d*)+(?i)<\head>`

展开 ▾

作者回复: 说一下几个问题:

1. 开头的 `(?i)` 已经表示正则是不区分大小写的了, 后面没必要再加一次。
2. `(\D*|\d*)+` 这种写法是非常不好的, 括号里面是0到多次, 然后后面又是一到多次, 效率比较低。

你可以在这里测试下, 你这个正则查找了2052步, <https://regex101.com/r/kJfvd6/2/>

我猜想你想表示 `[\d\D]+`, 改写成

`(?i)<head>[\d\D]+<\head>`

之后, 你会发现只需要查找23步。



