Ruby 面向对象 Ruby 数据库访问 – DBI 教程

Ruby 正则表达式

正则表达式是一种特殊序列的字符，它通过使用有专门语法的模式来匹配或查找字符串集合。

正则表达式用事先定义好的一些特定字符、及这些特定字符的组合，组成一个"规则字符串"，这个"规则字符串"用来表达对字符串的一种过滤逻辑。

语法

正则表达式从字面上看是一种介于斜杠之间或介于跟在 %r 后的任意分隔符之间的模式，如下所示：

```
/pattern/
/pattern/im # 可以指定选项
%r!/usr/local! # 使用分隔符的正则表达式
```

实例

```
#!/usr/bin/ruby
line1 = "Cats are smarter than dogs";
line2 = "Dogs also like meat";
if ( line1 =~ /Cats(.*)/ )
puts "Line1 contains Cats"
end
if ( line2 =~ /Cats(.*)/ )
puts "Line2 contains Dogs"
end
```

尝试一下 »

以上实例运行输出结果为：

```
Line1 contains Cats
```

正则表达式修饰符

正则表达式从字面上看可能包含一个可选的修饰符，用于控制各方面的匹配。修饰符在第二个斜杠字符后指定，如上面实例所示。下标列出了 可能的修饰符：

修饰符	描述
i	当匹配文本时忽略大小写。
o	只执行一次 #{} 插值，正则表达式在第一次时就进行判断。
x	忽略空格，允许在整个表达式中放入空白符和注释。

m	匹配多行，把换行字符识别为正常字符。
u,e,s,n	把正则表达式解释为 Unicode（UTF-8）、EUC、SJIS 或 ASCII。如果没有指定修饰符，则认为正则表达式使用的是源编码。

就像字符串通过 %Q 进行分隔一样，Ruby 允许您以 %r 作为正则表达式的开头，后面跟着任意分隔符。这在描述包含大量您不想转义的斜杠字符时非常有用。

```
# 下面匹配单个斜杠字符，不转义
%r|/|
# Flag 字符可通过下面的语法进行匹配
%r[</(.*)>]i
```

正则表达式模式

除了控制字符，**(+ ? . * ^ $ ( ) [ ] { } | \)**，其他所有字符都匹配本身。您可以通过在控制字符前放置一个反斜杠来对控制字符进行转义。

下表列出了 Ruby 中可用的正则表达式语法。

模式	描述
^	匹配行的开头。
$	匹配行的结尾。
.	匹配除了换行符以外的任意单字符。使用 m 选项时，它也可以匹配换行符。
[...]	匹配在方括号中的任意单字符。
[^...]	匹配不在方括号中的任意单字符。
re*	匹配前面的子表达式零次或多次。
re+	匹配前面的子表达式一次或多次。
re?	匹配前面的子表达式零次或一次。
re{ n}	匹配前面的子表达式 n 次。
re{ n,}	匹配前面的子表达式 n 次或 n 次以上。
re{ n, m}	匹配前面的子表达式至少 n 次至多 m 次。
a| b	匹配 a 或 b。
(re)	对正则表达式进行分组，并记住匹配文本。
(?imx)	暂时打开正则表达式内的 i、 m 或 x 选项。如果在圆括号中，则只影响圆括号内的部分。

(?-imx)	暂时关闭正则表达式内的 i、 m 或 x 选项。如果在圆括号中，则只影响圆括号内的部分。
(?: re)	对正则表达式进行分组，但不记住匹配文本。
(?imx: re)	暂时打开圆括号内的 i、 m 或 x 选项。
(?-imx: re)	暂时关闭圆括号内的 i、 m 或 x 选项。
(?#...)	注释。
(?= re)	使用模式指定位置。没有范围。
(?! re)	使用模式的否定指定位置。没有范围。
(?> re)	匹配无回溯的独立模式。
\w	匹配单词字符。
\W	匹配非单词字符。
\s	匹配空白字符。等价于 [\t\n\r\f]。
\S	匹配非空白字符。
\d	匹配数字。等价于 [0-9]。
\D	匹配非数字。
\A	匹配字符串的开头。
\Z	匹配字符串的结尾。如果存在换行符，则只匹配到换行符之前。
\z	匹配字符串的结尾。
\G	匹配最后一个匹配完成的点。
\b	当在括号外时匹配单词边界，当在括号内时匹配退格键（0x08）。
\B	匹配非单词边界。
\n, \t, etc.	匹配换行符、回车符、制表符，等等。
\1...\9	匹配第 n 个分组子表达式。
\10	如果已匹配过，则匹配第 n 个分组子表达式。否则指向字符编码的八进制表示。

正则表达式实例

字符

实例	描述
/ruby/	匹配 "ruby"
¥	匹配 Yen 符号。Ruby 1.9 和 Ruby 1.8 支持多个字符。

字符类

实例	描述
/[Rr]uby/	匹配 "Ruby" 或 "ruby"
/rub[ye]/	匹配 "ruby" 或 "rube"
/[aeiou]/	匹配任何一个小写元音字母
/[0-9]/	匹配任何一个数字，与 /[0123456789]/ 相同
/[a-z]/	匹配任何一个小写 ASCII 字母
/[A-Z]/	匹配任何一个大写 ASCII 字母
/[a-zA-Z0-9]/	匹配任何一个括号内的字符
/[^aeiou]/	匹配任何一个非小写元音字母的字符
/[^0-9]/	匹配任何一个非数字字符

特殊字符类

实例	描述
/./	匹配除了换行符以外的其他任意字符
/./m	在多行模式下，也能匹配换行符
/\d/	匹配一个数字，等同于 /[0-9]/
/\D/	匹配一个非数字，等同于 /[^0-9]/
/\s/	匹配一个空白字符，等同于 /[\t\r\n\f]/
/\S/	匹配一个非空白字符，等同于 /[^ \t\r\n\f]/

/\w/	匹配一个单词字符，等同于 /[A-Za-z0-9_]/
/\W/	匹配一个非单词字符，等同于 /[^A-Za-z0-9_]/

重复

实例	描述
/ruby?/	匹配 "rub" 或 "ruby"。其中，y 是可有可无的。
/ruby*/	匹配 "rub" 加上 0 个或多个的 y。
/ruby+/	匹配 "rub" 加上 1 个或多个的 y。
/\d{3}/	刚好匹配 3 个数字。
/\d{3,}/	匹配 3 个或多个数字。
/\d{3,5}/	匹配 3 个、4 个或 5 个数字。

非贪婪重复

这会匹配最小次数的重复。

实例	描述
/<.*>/	贪婪重复：匹配 "<ruby>perl>"
/<.*?>/	非贪婪重复：匹配 "<ruby>perl>" 中的 "<ruby>"

通过圆括号进行分组

实例	描述
/\D\d+/	无分组： + 重复 \d
/(\D\d)+/	分组： + 重复 \D\d 对
/([Rr]uby(,)?)+/	匹配 "Ruby"、"Ruby, ruby, ruby"，等等

反向引用

这会再次匹配之前匹配过的分组。

实例	描述
/([Rr])uby&\1ails/	匹配 ruby&rails 或 Ruby&Rails
/(["'])(?:(?!\1).)*\1/	单引号或双引号字符串。\1 匹配第一个分组所匹配的字符，\2 匹配第二个分组所匹配的字符，依此类推。

替换

实例	描述
/ruby|rube/	匹配 "ruby" 或 "rube"
/rub(y|le))/	匹配 "ruby" 或 "ruble"
/ruby(!+|\?)/	"ruby" 后跟一个或多个 ! 或者跟一个 ?

锚

这需要指定匹配位置。

实例	描述
/^Ruby/	匹配以 "Ruby" 开头的字符串或行
/Ruby$/	匹配以 "Ruby" 结尾的字符串或行
/\ARuby/	匹配以 "Ruby" 开头的字符串
/Ruby\Z/	匹配以 "Ruby" 结尾的字符串
/\bRuby\b/	匹配单词边界的 "Ruby"
/\brub\B/	\B 是非单词边界：匹配 "rube" 和 "ruby" 中的 "rub"，但不匹配单独的 "rub"
/Ruby(?=!)/	如果 "Ruby" 后跟着一个感叹号，则匹配 "Ruby"
/Ruby(?!!)/	如果 "Ruby" 后没有跟着一个感叹号，则匹配 "Ruby"

圆括号的特殊语法

实例	描述
/R(?#comment)/	匹配 "R"。所有剩余的字符都是注释。
/R(?i)uby/	当匹配 "uby" 时不区分大小写。
/R(?i:uby)/	与上面相同。
/rub(?:y|le))/	只分组，不进行 \1 反向引用

搜索和替换

sub 和 **gsub** 及它们的替代变量 **sub!** 和 **gsub!** 是使用正则表达式时重要的字符串方法。

所有这些方法都是使用正则表达式模式执行搜索与替换操作。**sub** 和 **sub!** 替换模式的第一次出现，**gsub** 和 **gsub!** 替换模式的所有出现。

sub 和 **gsub** 返回一个新的字符串，保持原始的字符串不被修改，而 **sub!** 和 **gsub!** 则会修改它们调用的字符串。

实例

```
#!/usr/bin/ruby
# -*- coding: UTF-8 -*-
phone = "138-3453-1111 #这是一个电话号码"
# 删除 Ruby 的注释
phone = phone.sub!(/#.*$/, "")
puts "电话号码 : #{phone}"
# 移除数字以外的其他字符
phone = phone.gsub!(/\D/, "")
puts "电话号码 : #{phone}"
```

尝试一下 »

以上实例运行输出结果为：

```
电话号码 : 138-3453-1111
电话号码 : 13834531111
```

实例

```
#!/usr/bin/ruby
# -*- coding: UTF-8 -*-
text = "rails 是 rails, Ruby on Rails 非常好的 Ruby 框架"
# 把所有的 "rails" 改为 "Rails"
text.gsub!("rails", "Rails")
# 把所有的单词 "Rails" 都改成首字母大写
text.gsub!(/\brails\b/, "Rails")
puts "#{text}"
```

尝试一下 »

以上实例运行输出结果为：

```
Rails 是 Rails,  Ruby on Rails 非常好的 Ruby 框架
```

← Ruby 面向对象 Ruby 数据库访问 – DBI 教程 →