正则表达式（regular expression）就是用一个“字符串”来描述一个特征，然后去验证另一个“字符串”是否符合这个特征。比如 表达式“ab+” 描述的特征是“一个 'a' 和 任意个 'b' ”，那么 'ab', 'abb', 'abbbbbbbbbb' 都符合这个特征。

    正则表达式可以用来：（1）验证字符串是否符合指定特征，比如验证是否是合法的邮件地址。（2）用来查找字符串，从一个长的文本中查找符合指定特征的字符串，比查找固定字符串更加灵活方便。（3）用来替换，比普通的替换更强大。

   虽然简化了，但是阅读性差。

#### **1. 正则表达式规则**

##### **1.1 普通字符**

    字母、数字、汉字、下划线、以及后边章节中没有特殊定义的标点符号，都是"普通字符"。表达式中的普通字符，在匹配一个字符串的时候，匹配与之相同的一个字符。  
  
    [举例1：表达式 "c"，在匹配字符串 "abcde" 时](http://www.regexlab.com/zh/workshop.asp?pat=c&txt=abcde" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到的内容是："c"；匹配到的位置是：开始于2，结束于3。（注：下标从0开始还是从1开始，因当前编程语言的不同而可能不同）  
  
    [举例2：表达式 "bcd"，在匹配字符串 "abcde" 时](http://www.regexlab.com/zh/workshop.asp?pat=bcd&txt=abcde" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到的内容是："bcd"；匹配到的位置是：开始于1，结束于4。

##### **1.2 简单的转义字符**

    一些不便书写的字符，采用在前面加 "\" 的方法。这些字符其实我们都已经熟知了。

|  |  |
| --- | --- |
| 表达式 | 可匹配 |
| \r, \n | 代表回车和换行符 |
| \t | 制表符 |
| \\ | 代表 "\" 本身 |

****还有其他一些在后边章节中有特殊用处的标点符号，在前面加 "\" 后，就代表该符号本身。比如：^, $ 都有特殊意义，如果要想匹配字符串中 "^" 和 "$" 字符，则表达式就需要写成 "\^" 和 "\$"。****

|  |  |
| --- | --- |
| 表达式 | 可匹配 |
| \^ | 匹配 ^ 符号本身 |
| \$ | 匹配 $ 符号本身 |
| \. | 匹配小数点（.）本身 |

    这些转义字符的匹配方法与 "普通字符" 是类似的。也是匹配与之相同的一个字符。  
  
    [举例1：表达式 "\$d"，在匹配字符串 "abc$de" 时](http://www.regexlab.com/zh/workshop.asp?pat=\\$d&txt=abc$de" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到的内容是："$d"；匹配到的位置是：开始于3，结束于5。

##### **1.3 能够与 '多种字符' 匹配的表达式**

    正则表达式中的一些表示方法，可以匹配 '多种字符' 其中的任意一个字符。比如，表达式 "\d" 可以匹配任意一个数字。虽然可以匹配其中任意字符，但是只能是一个，不是多个。这就好比玩扑克牌时候，大小王可以代替任意一张牌，但是只能代替一张牌。

|  |  |
| --- | --- |
| 表达式 | 可匹配 |
| ****\d**** | 任意一个数字，0~9 中的任意一个 |
| ****\w**** | 任意一个字母或数字或下划线，也就是 A~Z,a~z,0~9,\_ 中任意一个 |
| ****\s**** | 包括空格、制表符、换页符等空白字符的其中任意一个 |
| ****.**** | 小数点可以匹配除了换行符（\n）以外的任意一个字符 |

[举例1：表达式 "\d\d"，在匹配 "abc123" 时](http://www.regexlab.com/zh/workshop.asp?pat=\\d\\d&txt=abc123" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配的结果是：成功；匹配到的内容是："12"；匹配到的位置是：开始于3，结束于5。  
  
    [举例2：表达式 "a.\d"，在匹配 "aaa100" 时](http://www.regexlab.com/zh/workshop.asp?pat=a.\\d&txt=aaa100" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配的结果是：成功；匹配到的内容是："aa1"；匹配到的位置是：开始于1，结束于4。

##### **1.4 自定义能够匹配 '多种字符' 的表达式**

    使用方括号 [ ] 包含一系列字符，能够匹配其中任意一个字符。用 [^ ] 包含一系列字符，则能够匹配其中字符之外的任意一个字符。同样的道理，虽然可以匹配其中任意一个，但是只能是一个，不是多个。

|  |  |
| --- | --- |
| 表达式 | 可匹配 |
| ****[ab5@]**** | 匹配 "a" 或 "b" 或 "5" 或 "@" |
| ****[^abc]**** | 匹配 "a","b","c" 之外的任意一个字符 |
| ****[f-k]**** | 匹配 "f"~"k" 之间的任意一个字母 |
| ****[^A-F0-3]**** | 匹配 "A"~"F","0"~"3" 之外的任意一个字符 |

[举例1：表达式 "[bcd][bcd]" 匹配 "abc123" 时](http://www.regexlab.com/zh/workshop.asp?pat=[bcd][bcd]&txt=abc123" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配的结果是：成功；匹配到的内容是："bc"；匹配到的位置是：开始于1，结束于3。  
  
    [举例2：表达式 "[^abc]" 匹配 "abc123" 时](http://www.regexlab.com/zh/workshop.asp?pat=[^abc]&txt=abc123" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配的结果是：成功；匹配到的内容是："1"；匹配到的位置是：开始于3，结束于4。

##### **1.5 修饰匹配次数的特殊符号**

    前面章节中讲到的表达式，无论是只能匹配一种字符的表达式，还是可以匹配多种字符其中任意一个的表达式，都只能匹配一次。如果使用表达式再加上修饰匹配次数的特殊符号，那么不用重复书写表达式就可以重复匹配。  
  
    使用方法是：****"次数修饰"放在"被修饰的表达式"后边****。比如："[bcd][bcd]" 可以写成 "[bcd]{2}"。

|  |  |
| --- | --- |
| 表达式 | 作用 |
| ****{n}**** | 表达式重复n次，比如：["\w{2}" 相当于 "\w\w"](http://www.regexlab.com/zh/workshop.asp?pat=\\w{2}&txt=ab+c6" \t "http://blog.csdn.net/random0708/article/details/_blank)；["a{5}" 相当于 "aaaaa"](http://www.regexlab.com/zh/workshop.asp?pat=a{5}&txt=bbaaaaaddee" \t "http://blog.csdn.net/random0708/article/details/_blank) |
| ****{m,n}**** | 表达式至少重复m次，最多重复n次，比如：["ba{1,3}"可以匹配 "ba"或"baa"或"baaa"](http://www.regexlab.com/zh/workshop.asp?pat=ba{1,3}&txt=a,baaa,baa,b,ba" \t "http://blog.csdn.net/random0708/article/details/_blank) |
| ****{m,}**** | 表达式至少重复m次，比如：["\w\d{2,}"可以匹配 "a12","\_456","M12344"...](http://www.regexlab.com/zh/workshop.asp?pat=\\w\\d{2,}&txt=b1,a12,_456,_4AA,M12344,12346546547446534543543" \t "http://blog.csdn.net/random0708/article/details/_blank) |
| ****?**** | 匹配表达式0次或者1次，相当于 {0,1}，比如：["a[cd]?"可以匹配 "a","ac","ad"](http://www.regexlab.com/zh/workshop.asp?pat=a[cd]?&txt=a,c,d,ac,ad" \t "http://blog.csdn.net/random0708/article/details/_blank) |
| ****+**** | 表达式至少出现1次，相当于 {1,}，比如：["a+b"可以匹配 "ab","aab","aaab"...](http://www.regexlab.com/zh/workshop.asp?pat=a+b&txt=a,b,ab,aab,aaab" \t "http://blog.csdn.net/random0708/article/details/_blank) |
| ****\***** | 表达式不出现或出现任意次，相当于 {0,}，比如：["\^\*b"可以匹配 "b","^^^b"...](http://www.regexlab.com/zh/workshop.asp?pat=\\^*b&txt=^,b,^^^b,^^^^^^^b" \t "http://blog.csdn.net/random0708/article/details/_blank) |

[举例1：表达式 "\d+\.?\d\*" 在匹配 "It costs $12.5" 时](http://www.regexlab.com/zh/workshop.asp?pat=\\d+\\.?\\d*&txt=It costs $12.5" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配的结果是：成功；匹配到的内容是："12.5"；匹配到的位置是：开始于10，结束于14。  
  
    [举例2：表达式 "](http://www.regexlab.com/zh/workshop.asp?pat=go{2,8}gle&txt=Ads by goooooogle, or gooogle" \t "http://blog.csdn.net/random0708/article/details/_blank)**[Go](http://lib.csdn.net/base/go" \o "Go知识库" \t "http://blog.csdn.net/random0708/article/details/_blank)**{2,8}gle" 在匹配 "Ads by goooooogle" 时，匹配的结果是：成功；匹配到的内容是："goooooogle"；匹配到的位置是：开始于7，结束于17。

##### **1.6 其他一些代表抽象意义的特殊符号**

    一些符号在表达式中代表抽象的特殊意义：

|  |  |
| --- | --- |
| 表达式 | 作用 |
| ^ | 与字符串开始的地方匹配，不匹配任何字符 |
| $ | 与字符串结束的地方匹配，不匹配任何字符 |
| \b | 匹配一个单词边界，也就是单词和空格之间的位置，不匹配任何字符 |

    进一步的文字说明仍然比较抽象，因此，举例帮助大家理解。  
  
    [举例1：表达式 "^aaa" 在匹配 "xxx aaa xxx" 时](http://www.regexlab.com/zh/workshop.asp?pat=^aaa&txt=xxx+aaa+xxx" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：失败。因为 "^" 要求与字符串开始的地方匹配，因此，只有当 "aaa" 位于字符串的开头的时候，"^aaa" 才能匹配，[比如："aaa xxx xxx"](http://www.regexlab.com/zh/workshop.asp?pat=^aaa&txt=aaa+xxx+xxx" \t "http://blog.csdn.net/random0708/article/details/_blank)。  
  
    [举例2：表达式 "aaa$" 在匹配 "xxx aaa xxx" 时](http://www.regexlab.com/zh/workshop.asp?pat=aaa$&txt=xxx+aaa+xxx" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：失败。因为 "$" 要求与字符串结束的地方匹配，因此，只有当 "aaa" 位于字符串的结尾的时候，"aaa$" 才能匹配，[比如："xxx xxx aaa"](http://www.regexlab.com/zh/workshop.asp?pat=aaa$&txt=xxx+xxx+aaa" \t "http://blog.csdn.net/random0708/article/details/_blank)。  
  
    [举例3：表达式 ".\b." 在匹配 "@@@abc" 时](http://www.regexlab.com/zh/workshop.asp?pat=.\\b.&txt=@@@abc" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到的内容是："@a"；匹配到的位置是：开始于2，结束于4。  
    进一步说明："\b" 与 "^" 和 "$" 类似，本身不匹配任何字符，但是它要求它在匹配结果中所处位置的左右两边，其中一边是 "\w" 范围，另一边是 非"\w" 的范围。  
  
    [举例4：表达式 "\bend\b" 在匹配 "weekend,endfor,end" 时](http://www.regexlab.com/zh/workshop.asp?pat=\\bend\\b&txt=weekend,endfor,end" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到的内容是："end"；匹配到的位置是：开始于15，结束于18。

    一些符号可以影响表达式内部的子表达式之间的关系：

|  |  |
| --- | --- |
| 表达式 | 作用 |
| | | 左右两边表达式之间 "或" 关系，匹配左边或者右边 |
| ( ) | (1). 在被修饰匹配次数的时候，括号中的表达式可以作为整体被修饰 (2). 取匹配结果的时候，括号中的表达式匹配到的内容可以被单独得到 |

[举例5：表达式 "Tom|Jack" 在匹配字符串 "I'm Tom, he is Jack" 时](http://www.regexlab.com/zh/workshop.asp?pat=Tom|Jack&txt=I'm+Tom,+he+is+Jack" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到的内容是："Tom"；匹配到的位置是：开始于4，结束于7。匹配下一个时，匹配结果是：成功；匹配到的内容是："Jack"；匹配到的位置时：开始于15，结束于19。  
  
    [举例6：表达式 "(go\s\*)+" 在匹配 "Let's go go go!" 时](http://www.regexlab.com/zh/workshop.asp?pat=(go\\s*)+&txt=Let's go go go!" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到内容是："go go go"；匹配到的位置是：开始于6，结束于14。  
  
    [举例7：表达式 "￥(\d+\.?\d\*)" 在匹配 "＄10.9,￥20.5" 时](http://www.regexlab.com/zh/workshop.asp?pat=%uFFE5(\\d+\\.?\\d*)&txt=%uFF0410.9,%uFFE520.5" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配的结果是：成功；匹配到的内容是："￥20.5"；匹配到的位置是：开始于6，结束于10。单独获取括号范围匹配到的内容是："20.5"。

#### **2. 正则表达式中的一些高级规则**

##### **2.1 匹配次数中的贪婪与非贪婪**

    在使用修饰匹配次数的特殊符号时，有几种表示方法可以使同一个表达式能够匹配不同的次数，比如："{m,n}", "{m,}", "?", "\*", "+"，具体匹配的次数随被匹配的字符串而定。这种重复匹配不定次数的表达式在匹配过程中，总是尽可能多的匹配。比如，针对文本 "dxxxdxxxd"，举例如下：

|  |  |
| --- | --- |
| 表达式 | 匹配结果 |
| [(d)(\w+)](http://www.regexlab.com/zh/workshop.asp?pat=(d)(\\w+)&txt=dxxxdxxxd" \t "http://blog.csdn.net/random0708/article/details/_blank) | "\w+" 将匹配第一个 "d" 之后的所有字符 "xxxdxxxd" |
| [(d)(\w+)(d)](http://www.regexlab.com/zh/workshop.asp?pat=(d)(\\w+)(d)&txt=dxxxdxxxd" \t "http://blog.csdn.net/random0708/article/details/_blank) | "\w+" 将匹配第一个 "d" 和最后一个 "d" 之间的所有字符 "xxxdxxx"。虽然 "\w+" 也能够匹配上最后一个 "d"，但是为了使整个表达式匹配成功，"\w+" 可以 "让出" 它本来能够匹配的最后一个 "d" |

    由此可见，"\w+" 在匹配的时候，总是尽可能多的匹配符合它规则的字符。虽然第二个举例中，它没有匹配最后一个 "d"，但那也是为了让整个表达式能够匹配成功。同理，带 "\*" 和 "{m,n}" 的表达式都是尽可能地多匹配，带 "?" 的表达式在可匹配可不匹配的时候，也是尽可能的 "要匹配"。这 种匹配原则就叫作 "贪婪" 模式 。

    非贪婪模式：  
  
    在修饰匹配次数的特殊符号后再加上一个 "?" 号，则可以使匹配次数不定的表达式尽可能少的匹配，使可匹配可不匹配的表达式，尽可能的 "不匹配"。这种匹配原则叫作 "非贪婪" 模式，也叫作 "勉强" 模式。如果少匹配就会导致整个表达式匹配失败的时候，与贪婪模式类似，非贪婪模式会最小限度的再匹配一些，以使整个表达式匹配成功。举例如下，针对文本 "dxxxdxxxd" 举例：

|  |  |
| --- | --- |
| 表达式 | 匹配结果 |
| [(d)(\w+?)](http://www.regexlab.com/zh/workshop.asp?pat=(d)(\\w+?)&txt=dxxxdxxxd" \t "http://blog.csdn.net/random0708/article/details/_blank) | "\w+?" 将尽可能少的匹配第一个 "d" 之后的字符，结果是："\w+?" 只匹配了一个 "x" |
| [(d)(\w+?)(d)](http://www.regexlab.com/zh/workshop.asp?pat=(d)(\\w+?)(d)&txt=dxxxdxxxd" \t "http://blog.csdn.net/random0708/article/details/_blank) | 为了让整个表达式匹配成功，"\w+?" 不得不匹配 "xxx" 才可以让后边的 "d" 匹配，从而使整个表达式匹配成功。因此，结果是："\w+?" 匹配 "xxx" |

    更多的情况，举例如下：  
  
    [举例1：表达式 "<td>(.\*)</td>" 与字符串 "<td><p>aa</p></td> <td><p>bb</p></td>" 匹配时](http://www.regexlab.com/zh/workshop.asp?pat=<td>(.*)</td>&txt=<td><p>aa</p></td><td><p>bb</p></td>" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配的结果是：成功；匹配到的内容是 "<td><p>aa</p></td> <td><p>bb</p></td>" 整个字符串， 表达式中的 "</td>" 将与字符串中最后一个 "</td>" 匹配。   
  
    [举例2：相比之下，表达式 "<td>(.\*?)</td>" 匹配举例1中同样的字符串时](http://www.regexlab.com/zh/workshop.asp?pat=<td>(.*?)</td>&txt=<td><p>aa</p></td><td><p>bb</p></td>" \t "http://blog.csdn.net/random0708/article/details/_blank)，将只得到 "<td><p>aa</p></td>"， 再次匹配下一个时，可以得到第二个 "<td><p>bb</p></td>"。

##### **2.2 反向引用 \1, \2...**

    表达式在匹配时，表达式引擎会将小括号 "( )" 包含的表达式所匹配到的字符串记录下来。在获取匹配结果的时候，小括号包含的表达式所匹配到的字符串可以单独获取。这一点，在前面的举例中，已经多次展示了。在实际应用场合中，当用某种边界来查找，而所要获取的内容又不包含边界时，必须使用小括号来指定所要的范围。比如前面的 "<td>(.\*?)</td>"。  
  
    其实，"小括号包含的表达式所匹配到的字符串" 不仅是在匹配结束后才可以使用，在匹配过程中也可以使用。表达式后边的部分，可以引用前面 "括号内的子匹配已经匹配到的字符串"。引用方法是 "\" 加上一个数字。"\1" 引用第1对括号内匹配到的字符串，"\2" 引用第2对括号内匹配到的字符串……以此类推，如果一对括号内包含另一对括号，则外层的括号先排序号。换句话说，哪一对的左括号 "(" 在前，那这一对就先排序号。

    举例如下：  
  
    [举例1：表达式 "('|")(.\*?)(\1)" 在匹配 " 'Hello', "World" " 时](http://www.regexlab.com/zh/workshop.asp?pat=('|")(.*?)(\\1)&txt='Hello',+"World"" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到的内容是：" 'Hello' "。再次匹配下一个时，可以匹配到 " "World" "。  
  
    [举例2：表达式 "(\w)\1{4,}" 在匹配 "aa bbbb abcdefg ccccc 111121111 999999999" 时](http://www.regexlab.com/zh/workshop.asp?pat=(\\w)\\1{4,}&txt=aa bbbb abcdefg ccccc 111121111 999999999" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是：成功；匹配到的内容是 "ccccc"。再次匹配下一个时，将得到 999999999。这个表达式要求 "\w" 范围的字符至少重复5次，[注意与 "\w{5,}" 之间的区别](http://www.regexlab.com/zh/workshop.asp?pat=\\w{5,}&txt=aa bbbb abcdefg ccccc 111121111 999999999" \t "http://blog.csdn.net/random0708/article/details/_blank)。  
  
    [举例3：表达式 "<(\w+)\s\*(\w+(=('|").\*?\4)?\s\*)\*>.\*?</\1>" 在匹配 "<td id='td1' style="bgcolor:white"></td>" 时](http://www.regexlab.com/zh/workshop.asp?pat=<(\\w+)\\s*(\\w+(=('|").*?\\4)?\\s*)*>.*?</\\1>&txt=<td+id='td1'+style="bgcolor:white"></td>%0D%0A<body+onload="doit()"></body>" \t "http://blog.csdn.net/random0708/article/details/_blank)，匹配结果是成功。如果 "<td>" 与 "</td>" 不配对，则会匹配失败；如果改成其他配对，也可以匹配成功。

##### **2.3 预搜索，不匹配；反向预搜索，不匹配**

    前面的章节中，我讲到了几个代表抽象意义的特殊符号："^"，"$"，"\b"。它们都有一个共同点，那就是：它们本身不匹配任何字符，只是对 "字符串的两头" 或者 "字符之间的缝隙" 附加了一个条件。理解到这个概念以后，本节将继续介绍另外一种对 "两头" 或者 "缝隙" 附加条件的，更加灵活的表示方法。

    正向预搜索："(?=xxxxx)"，"(?!xxxxx)"  
  
    格式："(?=xxxxx)"，在被匹配的字符串中，它对所处的 "缝隙" 或者 "两头" 附加的条件是：所在缝隙的右侧，必须能够匹配上 xxxxx 这部分的表达式。因为它只是在此作为这个缝隙上附加的条件，所以它并不影响后边的表达式去真正匹配这个缝隙之后的字符。这就类似 "\b"，本身不匹配任何字符。"\b" 只是将所在缝隙之前、之后的字符取来进行了一下判断，不会影响后边的表达式来真正的匹配。  
  
    [举例1：表达式 "Windows (?=NT|XP)" 在匹配 "Windows 98, Windows NT, Windows 2000" 时](http://www.regexlab.com/zh/workshop.asp?pat=Windows+(?=NT|XP)&txt=Windows+98,+Windows+NT,+Windows+2000" \t "http://blog.csdn.net/random0708/article/details/_blank)，将只匹配 "Windows NT" 中的 "Windows "，其他的 "Windows " 字样则不被匹配。  
  
    [举例2：表达式 "(\w)((?=\1\1\1)(\1))+" 在匹配字符串 "aaa ffffff 999999999" 时](http://www.regexlab.com/zh/workshop.asp?pat=(\\w)((?=\\1\\1\\1)(\\1))+&txt=aaa+ffffff+999999999" \t "http://blog.csdn.net/random0708/article/details/_blank)，将可以匹配6个"f"的前4个，可以匹配9个"9"的前7个。这个表达式可以读解成：重复4次以上的字母数字，则匹配其剩下最后2位之前的部分。当然，这个表达式可以不这样写，在此的目的是作为演示之用。

    格式："(?!xxxxx)"，所在缝隙的右侧，必须不能匹配 xxxxx 这部分表达式。  
  
    [举例3：表达式 "((?!\bstop\b).)+" 在匹配 "fdjka ljfdl stop fjdsla fdj" 时](http://www.regexlab.com/zh/workshop.asp?pat=((?!\\bstop\\b).)+&txt=fdjka+ljfdl+stop+fjdsla+fdj" \t "http://blog.csdn.net/random0708/article/details/_blank)，将从头一直匹配到 "stop" 之前的位置，如果字符串中没有 "stop"，则匹配整个字符串。  
  
    [举例4：表达式 "do(?!\w)" 在匹配字符串 "done, do, dog" 时](http://www.regexlab.com/zh/workshop.asp?pat=do(?!\\w)&txt=done,+do,+dog" \t "http://blog.csdn.net/random0708/article/details/_blank)，只能匹配 "do"。在本条举例中，"do" 后边使用 "(?!\w)" 和使用 "\b" 效果是一样的。

    反向预搜索："(?<=xxxxx)"，"(?<!xxxxx)"  
  
    这两种格式的概念和正向预搜索是类似的，反向预搜索要求的条件是：所在缝隙的 "左侧"，两种格式分别要求必须能够匹配和必须不能够匹配指定表达式，而不是去判断右侧。与 "正向预搜索" 一样的是：它们都是对所在缝隙的一种附加条件，本身都不匹配任何字符。  
  
    举例5：表达式 "(?<=\d{4})\d+(?=\d{4})" 在匹配 "1234567890123456" 时，将匹配除了前4个数字和后4个数字之外的中间8个数字。由于 JScript.RegExp 不支持反向预搜索，因此，本条举例不能够进行演示。很多其他的引擎可以支持反向预搜索，比如：**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "http://blog.csdn.net/random0708/article/details/_blank)**1.4 以上的 **[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "http://blog.csdn.net/random0708/article/details/_blank)**.util.regex 包，.NET 中System.Text.RegularExpressions 命名空间，以及本站推荐的[最简单易用的 DEELX 正则引擎](http://www.regexlab.com/zh/deelx/" \t "http://blog.csdn.net/random0708/article/details/_blank)。

#### **3. 其他通用规则**

    还有一些在各个正则表达式引擎之间比较通用的规则，在前面的讲解过程中没有提到。

3.1 表达式中，可以使用 "\xXX" 和 "\uXXXX" 表示一个字符（"X" 表示一个十六进制数）

|  |  |
| --- | --- |
| 形式 | 字符范围 |
| \xXX | 编号在 0 ~ 255 范围的字符，比如：[空格可以使用 "\x20" 表示](http://www.regexlab.com/zh/workshop.asp?pat=\\x20&txt=It+is." \t "http://blog.csdn.net/random0708/article/details/_blank) |
| \uXXXX | 任何字符可以使用 "\u" 再加上其编号的4位十六进制数表示，比如：["\u4E2D"](http://www.regexlab.com/zh/workshop.asp?pat=\\u4E2D&txt=%D6%D0%B9%FA" \t "http://blog.csdn.net/random0708/article/details/_blank) |

3.2 在表达式 "\s"，"\d"，"\w"，"\b" 表示特殊意义的同时，对应的大写字母表示相反的意义

|  |  |
| --- | --- |
| 表达式 | 可匹配 |
| \S | [匹配所有非空白字符（"\s" 可匹配各个空白字符）](http://www.regexlab.com/zh/workshop.asp?pat=\\S+&txt=abc+123+@%23$%" \t "http://blog.csdn.net/random0708/article/details/_blank) |
| \D | [匹配所有的非数字字符](http://www.regexlab.com/zh/workshop.asp?pat=\\D+&txt=abc+123+@%23$%" \t "http://blog.csdn.net/random0708/article/details/_blank) |
| \W | [匹配所有的字母、数字、下划线以外的字符](http://www.regexlab.com/zh/workshop.asp?pat=\\W+&txt=abc+123+@%23$%" \t "http://blog.csdn.net/random0708/article/details/_blank) |
| \B | [匹配非单词边界，即左右两边都是 "\w" 范围或者左右两边都不是 "\w" 范围时的字符缝隙](http://www.regexlab.com/zh/workshop.asp?pat=\\B.\\B&txt=abc+123+@%23$%" \t "http://blog.csdn.net/random0708/article/details/_blank) |

3.3 在表达式中有特殊意义，需要添加 "\" 才能匹配该字符本身的字符汇总

|  |  |
| --- | --- |
| 字符 | 说明 |
| ^ | 匹配输入字符串的开始位置。要匹配 "^" 字符本身，请使用 "\^" |
| $ | 匹配输入字符串的结尾位置。要匹配 "$" 字符本身，请使用 "\$" |
| ( ) | 标记一个子表达式的开始和结束位置。要匹配小括号，请使用 "" |
| [ ] | 用来自定义能够匹配 '多种字符' 的表达式。要匹配中括号，请使用 "  " |
| { } | 修饰匹配次数的符号。要匹配大括号，请使用 "\{" 和 "\}" |
| . | 匹配除了换行符（\n）以外的任意一个字符。要匹配小数点本身，请使用 "\." |
| ? | 修饰匹配次数为 0 次或 1 次。要匹配 "?" 字符本身，请使用 "\?" |
| + | 修饰匹配次数为至少 1 次。要匹配 "+" 字符本身，请使用 "\+" |
| \* | 修饰匹配次数为 0 次或任意次。要匹配 "\*" 字符本身，请使用 "\\*" |
| | | 左右两边表达式之间 "或" 关系。匹配 "|" 本身，请使用 "\|" |

3.4 括号 "( )" 内的子表达式，如果希望匹配结果不进行记录供以后使用，可以使用 "(?:xxxxx)" 格式

[举例1：表达式 "(?:(\w)\1)+" 匹配 "a bbccdd efg" 时](http://www.regexlab.com/zh/workshop.asp?pat=(?:(\\w)\\1)+&txt=a bbccdd efg" \t "http://blog.csdn.net/random0708/article/details/_blank)，结果是 "bbccdd"。括号 "(?:)" 范围的匹配结果不进行记录，因此 "(\w)" 使用 "\1" 来引用。

3.5 常用的表达式属性设置简介：Ignorecase，Singleline，Multiline，Global

|  |  |
| --- | --- |
| 表达式属性 | 说明 |
| Ignorecase | 默认情况下，表达式中的字母是要区分大小写的。配置为 Ignorecase 可使匹配时不区分大小写。有的表达式引擎，把 "大小写" 概念延伸至 UNICODE 范围的大小写。 |
| Singleline | 默认情况下，小数点 "." 匹配除了换行符（\n）以外的字符。配置为 Singleline 可使小数点可匹配包括换行符在内的所有字符。 |
| Multiline | 默认情况下，表达式 "^" 和 "$" 只匹配字符串的开始 ① 和结尾 ④ 位置。如：  ①xxxxxxxxx②\n ③xxxxxxxxx④  配置为 Multiline 可以使 "^" 匹配 ① 外，还可以匹配换行符之后，下一行开始前 ③ 的位置，使 "$" 匹配 ④ 外，还可以匹配换行符之前，一行结束 ② 的位置。 |
| Global | 主要在将表达式用来替换时起作用，配置为 Global 表示替换所有的匹配。 |