**正则表达式语法**

在其他语言中，\\ 表示：我想要在正则表达式中插入一个普通的（字面上的）反斜杠，请不要给它任何特殊的意义。  
  
在 Java 中，\\ 表示：我要插入一个正则表达式的反斜线，所以其后的字符具有特殊的意义。  
  
所以，在其他的语言中（如 Perl），一个反斜杠 \ 就足以具有转义的作用，而在 Java 中正则表达式中则需要有两个反斜杠才能被解析为其他语言中的转义作用。也可以简单的理解在 Java 的正则表达式中，两个 \\ 代表其他语言中的一个 \，这也就是为什么表示一位数字的正则表达式是 \\d，而表示一个普通的反斜杠是 \\。  
  
System.out.print("\\");    // 输出为 \  
System.out.print("\\\\");  // 输出为 \\  
字符  
  
说明  
  
\  
  
将下一字符标记为特殊字符、文本、反向引用或八进制转义符。例如， n匹配字符 n。\n 匹配换行符。序列 \\\\ 匹配 \\ ，\\( 匹配 (。  
  
^  
  
匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性，^ 还会与"\n"或"\r"之后的位置匹配。  
  
$  
  
匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，$ 还会与"\n"或"\r"之前的位置匹配。  
  
\*  
  
零次或多次匹配前面的字符或子表达式。例如，zo\* 匹配"z"和"zoo"。\* 等效于 {0,}。  
  
+  
  
一次或多次匹配前面的字符或子表达式。例如，"zo+"与"zo"和"zoo"匹配，但与"z"不匹配。+ 等效于 {1,}。  
  
?  
  
零次或一次匹配前面的字符或子表达式。例如，"do(es)?"匹配"do"或"does"中的"do"。? 等效于 {0,1}。  
  
{n}  
  
n 是非负整数。正好匹配 n 次。例如，"o{2}"与"Bob"中的"o"不匹配，但与"food"中的两个"o"匹配。  
  
{n,}  
  
n 是非负整数。至少匹配 n 次。例如，"o{2,}"不匹配"Bob"中的"o"，而匹配"foooood"中的所有 o。"o{1,}"等效于"o+"。"o{0,}"等效于"o\*"。  
  
{n,m}  
  
m 和 n 是非负整数，其中 n <= m。匹配至少 n 次，至多 m 次。例如，"o{1,3}"匹配"fooooood"中的头三个 o。'o{0,1}' 等效于 'o?'。注意：您不能将空格插入逗号和数字之间。  
  
?  
  
当此字符紧随任何其他限定符（\*、+、?、{n}、{n,}、{n,m}）之后时，匹配模式是"非贪心的"。"非贪心的"模式匹配搜索到的、尽可能短的字符串，而默认的"贪心的"模式匹配搜索到的、尽可能长的字符串。例如，在字符串"oooo"中，"o+?"只匹配单个"o"，而"o+"匹配所有"o"。  
  
.  
  
匹配除"\r\n"之外的任何单个字符。若要匹配包括"\r\n"在内的任意字符，请使用诸如"[\s\S]"之类的模式。  
  
(pattern)  
  
匹配 pattern 并捕获该匹配的子表达式。可以使用 $0…$9 属性从结果"匹配"集合中检索捕获的匹配。若要匹配括号字符 ( )，请使用"\("或者"\)"。  
  
(?:pattern)  
  
匹配 pattern 但不捕获该匹配的子表达式，即它是一个非捕获匹配，不存储供以后使用的匹配。这对于用"or"字符 (|) 组合模式部件的情况很有用。例如，'industr(?:y|ies) 是比 'industry|industries' 更经济的表达式。  
  
(?=pattern)  
  
执行正向预测先行搜索的子表达式，该表达式匹配处于匹配 pattern 的字符串的起始点的字符串。它是一个非捕获匹配，即不能捕获供以后使用的匹配。例如，'Windows (?=95|98|NT|2000)' 匹配"Windows 2000"中的"Windows"，但不匹配"Windows 3.1"中的"Windows"。预测先行不占用字符，即发生匹配后，下一匹配的搜索紧随上一匹配之后，而不是在组成预测先行的字符后。  
  
(?!pattern)  
  
执行反向预测先行搜索的子表达式，该表达式匹配不处于匹配 pattern 的字符串的起始点的搜索字符串。它是一个非捕获匹配，即不能捕获供以后使用的匹配。例如，'Windows (?!95|98|NT|2000)' 匹配"Windows 3.1"中的 "Windows"，但不匹配"Windows 2000"中的"Windows"。预测先行不占用字符，即发生匹配后，下一匹配的搜索紧随上一匹配之后，而不是在组成预测先行的字符后。  
  
x|y  
  
匹配 x 或 y。例如，'z|food' 匹配"z"或"food"。'(z|f)ood' 匹配"zood"或"food"。  
  
[xyz]  
  
字符集。匹配包含的任一字符。例如，"[abc]"匹配"plain"中的"a"。  
  
[^xyz]  
  
反向字符集。匹配未包含的任何字符。例如，"[^abc]"匹配"plain"中"p"，"l"，"i"，"n"。  
  
[a-z]  
  
字符范围。匹配指定范围内的任何字符。例如，"[a-z]"匹配"a"到"z"范围内的任何小写字母。  
  
[^a-z]  
  
反向范围字符。匹配不在指定的范围内的任何字符。例如，"[^a-z]"匹配任何不在"a"到"z"范围内的任何字符。  
  
\b  
  
匹配一个字边界，即字与空格间的位置。例如，"er\b"匹配"never"中的"er"，但不匹配"verb"中的"er"。  
  
\B  
  
非字边界匹配。"er\B"匹配"verb"中的"er"，但不匹配"never"中的"er"。  
  
\cx  
  
匹配 x 指示的控制字符。例如，\cM 匹配 Control-M 或回车符。x 的值必须在 A-Z 或 a-z 之间。如果不是这样，则假定 c 就是"c"字符本身。  
  
\d  
  
数字字符匹配。等效于 [0-9]。  
  
\D  
  
非数字字符匹配。等效于 [^0-9]。  
  
\f  
  
换页符匹配。等效于 \x0c 和 \cL。  
  
\n  
  
换行符匹配。等效于 \x0a 和 \cJ。  
  
\r  
  
匹配一个回车符。等效于 \x0d 和 \cM。  
  
\s  
  
匹配任何空白字符，包括空格、制表符、换页符等。与 [ \f\n\r\t\v] 等效。  
  
\S  
  
匹配任何非空白字符。与 [^ \f\n\r\t\v] 等效。  
  
\t  
  
制表符匹配。与 \x09 和 \cI 等效。  
  
\v  
  
垂直制表符匹配。与 \x0b 和 \cK 等效。  
  
\w  
  
匹配任何字类字符，包括下划线。与"[A-Za-z0-9\_]"等效。  
  
\W  
  
与任何非单词字符匹配。与"[^A-Za-z0-9\_]"等效。  
  
\xn  
  
匹配 n，此处的 n 是一个十六进制转义码。十六进制转义码必须正好是两位数长。例如，"\x41"匹配"A"。"\x041"与"\x04"&"1"等效。允许在正则表达式中使用 ASCII 代码。  
  
\num  
  
匹配 num，此处的 num 是一个正整数。到捕获匹配的反向引用。例如，"(.)\1"匹配两个连续的相同字符。  
  
\n  
  
标识一个八进制转义码或反向引用。如果 \n 前面至少有 n 个捕获子表达式，那么 n 是反向引用。否则，如果 n 是八进制数 (0-7)，那么 n 是八进制转义码。  
  
\nm  
  
标识一个八进制转义码或反向引用。如果 \nm 前面至少有 nm 个捕获子表达式，那么 nm 是反向引用。如果 \nm 前面至少有 n 个捕获，则 n 是反向引用，后面跟有字符 m。如果两种前面的情况都不存在，则 \nm 匹配八进制值 nm，其中 n 和 m 是八进制数字 (0-7)。  
  
\nml  
  
当 n 是八进制数 (0-3)，m 和 l 是八进制数 (0-7) 时，匹配八进制转义码 nml。  
  
\un  
  
匹配 n，其中 n 是以四位十六进制数表示的 Unicode 字符。例如，\u00A9 匹配版权符号 (IMG_256)。  
  
根据 Java Language Specification 的要求，Java 源代码的字符串中的反斜线被解释为 Unicode 转义或其他字符转义。因此必须在字符串字面值中使用两个反斜线，表示正则表达式受到保护，不被 Java 字节码编译器解释。例如，当解释为正则表达式时，字符串字面值 "\b" 与单个退格字符匹配，而 "\\b" 与单词边界匹配。字符串字面值 "\(hello\)" 是非法的，将导致编译时错误；要与字符串 (hello) 匹配，必须使用字符串字面值 "\\(hello\\)"。