正则表达式（regular expressions）是一种描述字符串集的方法，它是以字符串集中各字符串的共有特征为依据的。

正则表达式可以用于搜索、编辑或者是操作文本和数据。

java.util.regex API 所支持的正则表达式语法

1. **字符串**

如果正则表达式是foo，输入的字符串也是 foo，这个匹配将会是成功的

1. **元字符**

API 所支持的元字符有：([{\^-$|}])?\*+.

1. **字符类**

|  |  |
| --- | --- |
| **字符类** | |
| [abc] | a, b 或 c（简单类） |
| [^abc] | 除 a, b 或 c 之外的任意字符（取反） |
| [a-zA-Z] | a 到 z，或 A 到 Z，包括（范围） |
| [a-d[m-p]] | a 到 d，或 m 到 p：[a-dm-p]（并集） |
| [a-z&&[def]] | d，e 或 f（交集） |
| [a-z&&[^bc]] | 除 b 和 c 之外的 a 到 z 字符：[ad-z]（差集） |
| [a-z&&[^m-p]] | a 到 z，并且不包括 m 到 p：[a-lq-z]（差集） |

**4　预定义字符类**

|  |  |
| --- | --- |
| **预定义字符类** | |
| . | 任何字符（匹配或者不匹配行结束符） |
| \d | 数字字符：[0-9] |
| \D | 非数字字符：[^0-9] |
| \s | 空白字符：[\t\n\x0B\f\r] |
| \S | 非空白字符：[^\s] |
| \w | 单词字符：[a-zA-Z\_0-9] |
| \W | 非单词字符：[^\w] |

**5　量词**

| **量　词　种　类** | | | **意　　义** |
| --- | --- | --- | --- |
| **贪婪** | **勉强** | **侵占** |
| X? | X?? | X?+ | 匹配 X 零次或一次 |
| X\* | X\*? | X\*+ | 匹配 X 零次或多次 |
| X+ | X+? | X++ | 匹配 X 一次或多次 |
| X{n} | X{n}? | X{n}+ | 匹配 X n 次 |
| X{n,} | X{n,}? | X{n,}+ | 匹配 X 至少 n 次 |
| X{n,m} | X{n,m}? | X{n,m}+ | 匹配 X 至少 n 次，但不多于 m 次 |

**2、贪婪、勉强和侵占量词间的不同**

在贪婪、勉强和侵占三个量词间有着细微的不同。  
　　贪婪量词之所以称之为“贪婪的”，这是由于它们强迫匹配器读入（或者称之为吃掉）整个输入的字符串，来优先尝试第一次匹配，如果第一次尝试匹配（对于整个输入的字符串）失败，匹配器会通过回退整个字符串的一个字符再一次进行尝试，不断地进行处理直到找到一个匹配，或者左边没有更多的字符来用于回退了。赖于在表达式中使用的量词，最终它将尝试地靠着 1 或 0 个字符的匹配。  
　　但是，勉强量词采用相反的途径：从输入字符串的开始处开始，因此每次勉强地吞噬一个字符来寻找匹配，最终它们会尝试整个输入的字符串。  
　　最后，侵占量词始终是吞掉整个输入的字符串，尝试着一次（仅有一次）匹配。不像贪婪量词那样，侵占量词绝不会回退，即使这样做是允许全部的匹配成功。