正则表达式(Regular Expression)

- 工具
  - RegexBuddy
- 语法
  - 字符
    - 普通字符：字母、数字、汉字、下划线，匹配与之相同的一个字符
    - 简单转义字符：\n（换行）,\t（制表）,\\（\本身）和 \^...（\^等有特殊作用的符号如要匹配自己的话要用转义）
    - 标准字符集合

      注意区分大小写，大写是相反的意思，匹配相反是不匹配
      - \d

        任意一个数字，0~9
      - \w

        任意一个字母、数字、汉字或下划线，A~Z、a~z、0~9、_和任意一个汉字
      - \s

        任意空白符，包括空格、制表符、换行符
      - .

        小数点可以匹配任意一个字符，换行除外（如果要匹配包括"\n"在内的所有字符，一般用[\s\S]）
    - 自定义字符集合

      [ ]方括号匹配方式，能够匹配方括号中的任意一个字符,^表示取反
      - [ab5@]

        匹配"a"或"b"或"5"或"@"
      - [^abc]

        匹配a、b、c之外的任意字符
      - [f-k]

        匹配“f"到"k"之间的字符
      - [^A-F0-3]

        匹配“A"-"F","0"-"3"之外的任意一个字符
  - 量词(Quantifier)

    修饰前面的一个表达式，如果要修饰多个表达式，就用( )把表达式包起来
    - {n}

      表达式重复n次
    - {m,n}

      表达式至少重复m次，最多重复n次
      - 贪婪模式 （默认）

        匹配符合的最长的字符串
      - 非贪婪模式 （在量词后面加？例：{m,n}? ）

        匹配符合的最短的字符串
    - {m,}

      表达式至少重复m次
    - ?

匹配表达式0或1次，相当于{0,1}

- +

表达式至少出现一次，相当于{1,}

- *

表达式不出现或出现任意次，相当于{0,}

- 字符边界

零宽：匹配的不是字符而是位置，符合某种条件的位置

- ^

与字符串开始的地方匹配

- $

与字符串结束的地方匹配

- \b

匹配一个单词的边界，当前位置前面的字符和后面的字符不全是\w

- 预搜索（零宽断言、环视）

零宽：匹配的不是字符而是位置，符合某种条件的位置

- (?=exp)

断言自身出现的位置的后面能匹配表达式exp

- (?!exp)

断言自身出现的位置的后面不能匹配表达式exp

- (?<=exp)

断言自身出现的位置的前面能匹配表达式exp

- (?<!exp)

断言自身出现的位置的前面不能匹配表达式exp

- 匹配模式

对文本的处理方式

- IGNORECASE 忽略大小写模式
  - 匹配时忽略大小写
  - 默认是区分大小写的
- SINGLELINE 单行模式
  - 整个文本看作一个字符串，只有一个开头一个结尾
  - 使小数点"."可以匹配包含换行符(\n)在内的任意字符
- MULTILINE 多行模式
  - 每行都是一个字符串
  - 在多行模式下，如果需要仅匹配字符串开始和结束位置，可以使用\A和\Z

- 选择符和分组

分支结构、捕获组合非捕获组

- | 分支结构

左右表达式之间“或”关系，匹配左边或右边

- ( ) 捕获组
  - （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
  - （3）、每一对括号会分配一个编号，使用（）的捕获根据左括号的顺序从1开始自动编号。捕获编号为零的第一个捕获是整个正则表达式模式匹配的文本
  - 反向引用：通过反向引用，可以对分组已捕获的字符串进行引用。

- (?:Expression) 非捕获组

  一些表达式中，不得不使用（），但又不需要保存（）中子表达式匹配的内容，这时可以用非捕获组来抵消（）带来的副作用。