1. 元字符

|  |  |
| --- | --- |
| 元字符 | 说明 |
| . | 匹配除换行符以外的任意字符 |
| \w | 匹配字母或数字或下划线或汉字 |
| \s | 匹配任意的空白字符 |
| \d | 匹配数字 |
| \b | 匹配单词的开始或结束 |
| ^ | 匹配字符串的开始 |
| $ | 匹配字符串的结束 |

1. 限定符

|  |  |
| --- | --- |
| 限定符 | 说明 |
| \* | 重复0次或更多次 |
| + | 重复一次或更多次 |
| ? | 重复0词或一次 |
| {n} | 重复n次 |
| {n, } | 重复n次或更多次 |
| {n, m} | 重复n到m次 |

1. 使用 [] 指定一个字符范围，如[0-6abcd]
2. 分枝条件：使用 | 分割
3. 使用 () 可以分组， 如 ((2[0-4]\d|25[0-5]|[01]?\d\d?)\.){3}(2[0-4]\d|25[0-5]|[01]?\d\d?) ip地址匹配
4. 反义

|  |  |
| --- | --- |
| 代码 | 说明 |
| \W | 匹配任意不是字母，数字，下划线，汉字的字符 |
| \S | 匹配任意不是空白符的字符 |
| \D | 匹配任意非数字的字符 |
| \B | 匹配不是单词开头或结束的位置 |
| [^x] | 匹配除了x以外的任意字符 |
| [^aeiou] | 匹配除了aeiou这几个字母以外的任意字符 |

1. 捕获分组

使用小括号指定一个子表达式后，匹配这个子表达式的文本，每个分组会有一个组号，规则是从左向右以分组的左括号为标志，第一个出现的分组的组号是1，第二个是2...

1. 后向引用

用于重复搜索前面的某个分组匹配的文本，\b(\w+)\b\s+\1\b可以用来匹配重复的单词，\1代表第一个分组匹配的文本

也可以指定分组名，语法：(?<Word>\w+) 或者 (?’word’\w+)，要反向引用使用\k<Word>

1. 常用分组语法

|  |  |  |
| --- | --- | --- |
| 分类 | 代码 | 说明 |
| 捕获 | (exp) | 匹配exp，并捕获文本到自动命名的组里 |
| (?<name>exp) | 匹配exp，并捕获文本到名称为name的组里 |
| (?:exp) | 匹配exp，不捕获文本，也不分配组号 |
| 零宽断言 | (?=exp) | 匹配exp前面的位置 |
| (?<=exp) | 匹配exp后面的位置 |
| (!exp) | 匹配后面跟的不是exp的位置 |
| (?<!exp) | 匹配前面不是exp的位置 |
| 注释 | (?#comment) | 注释 |

1. 懒惰：当正在表达式中包含能够接受重复的限定符时，通常是匹配尽可能多的字符，而懒惰是匹配尽量少的字符

|  |  |
| --- | --- |
| 代码 | 语法 |
| \*? | 重复任意次，但尽可能少重复 |
| +? | 重复1次或更多次，但尽可能少重复 |
| ?? | 重复0次或1次，但尽可能少重复 |
| {n,m}? | 重复n到m次，但尽可能少重复 |
| {n, }? | 重复n次以上，但尽可能少重复 |

1. 平衡组/递归匹配

(?'group') 把捕获的内容命名为group,并压入**堆栈(Stack)**

(?'-group') 从堆栈上弹出最后压入堆栈的名为group的捕获内容，如果堆栈本来为空，则本分组的匹配失败

(?(group)yes|no) 如果堆栈上存在以名为group的捕获内容的话，继续匹配yes部分的表达式，否则继续匹配no部分

(?!) 零宽负向先行断言，由于没有后缀表达式，试图匹配总是失败

< #最外层的左括号

[^<>]\* #最外层的左括号后面的不是括号的内容

(

(

(?'Open'<) #碰到了左括号，在黑板上写一个"Open"

[^<>]\* #匹配左括号后面的不是括号的内容

)+

(

(?'-Open'>) #碰到了右括号，擦掉一个"Open"

[^<>]\* #匹配右括号后面不是括号的内容

)+

)\*

(?(Open)(?!)) #在遇到最外层的右括号前面，判断黑板上还有没有没擦掉的"Open"；如果还有，则匹配失败

> #最外层的右括号

例子：<div[^>]\*>[^<>]\*(((?'Open'<div[^>]\*>)[^<>]\*)+((?'-Open'</div>)[^<>]\*)+)\*(?(Open)(?!))</div>.