### 正则表达式入门以及在iUAP DI中的使用

目录

[正则表达式入门以及在iUAP DI中的使用 1](#_Toc417462197)

[1. 什么是正则表达式 1](#_Toc417462198)

[2. 正则表达式规则 1](#_Toc417462199)

[2.1. 普通字符 1](#_Toc417462200)

[2.2. 简单的转义字符 1](#_Toc417462201)

[2.3. 能够与 '多种字符' 匹配的表达式 2](#_Toc417462202)

[2.4. 自定义能够匹配 '多种字符' 的表达式 2](#_Toc417462203)

[2.5. 修饰匹配次数的特殊符号 3](#_Toc417462204)

[2.6. 其他代表抽象意义的特殊符号 4](#_Toc417462205)

[2.7. 字符大小写与贪婪模式 5](#_Toc417462206)

[**3. 在iUAP DI中正则表达式的使用 5**](#_Toc417462207)

[**3.1. 应用正则表达式的转换/作业项 5**](#_Toc417462208)

[**3.2. 应用场景及案例 6**](#_Toc417462209)

[**3.2.1. 案例一：匹配和过滤文件名 6**](#_Toc417462210)

[**3.2.2. 案例二：过滤数据 6**](#_Toc417462211)

[**3.2.3. 案例三：数据检验 7**](#_Toc417462212)

[**3.2.4. 案例四：过滤表名 7**](#_Toc417462213)

[4. 总结 8](#_Toc417462214)

# 什么是正则表达式

 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。

# 正则表达式规则

## 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，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，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。

## 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。

|  |  |
| --- | --- |
| 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$de" 时，匹配结果是：成功；匹配到的内容是："$d"；匹配到的位置是：开始于3，结束于5。

## 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。

|  |  |
| --- | --- |
| 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，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，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。

## 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，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，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。

## 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"； "a{5}" 相当于 "aaaaa" |
| {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" |
| {m,}  | 表达式至少重复m次，比如： "\w\d{2,}"可以匹配 "a12","\_456","M12344"... |
| ?  | 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" |
| +  | 表达式至少出现1次，相当于 {1,}，比如： "a+b"可以匹配 "ab","aab","aaab"... |
| \*  | 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"... |

 举例1：表达式 "\d+\.?\d\*" 在匹配 "It costs $12.5" 时，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 "go{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。

## 其他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 |

举例1：表达式 "^aaa" 在匹配 "xxx aaa xxx" 时，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"。

举例2：表达式 "aaa$" 在匹配 "xxx aaa xxx" 时，匹配结果是：失败。因为 "$" 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"。

举例3：表达式 ".\b." 在匹配 "@@@abc" 时，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，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，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，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，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"。

## 字符大小写与贪婪模式

正则表达式一般来说是区分大小写的，对于字符串”AAAbcdddd”,使用正则表达式”aaa\w+”是无法匹配，但有时候需要不区分大小写匹配，例如在匹配数据库表名时。可以在组前添加特殊的非捕捉构造(?)实现,在上例中使用正则表达式”(?i)aaa\w+”就可以匹配字符串”AAAbcdddd”。

当正则表达式中包含能接受重复的限定符时，通常的行为是（在使整个表达式能得到匹配的前提下）匹配尽可能多的字符。对于正则表达式”a.\*b”，它将会匹配最长的以a开始，以b结束的字符串。如果用它来搜索aabab的话，它会匹配整个字符串aabab。这被称为贪婪匹配。有时，我们更需要懒惰匹配，也就是匹配尽可能少的字符。前面给出的限定符都可以被转化为懒惰匹配模式，只要在它后面加上一个问号?。这样”.\*?”就意味着在能使整个匹配成功的前提下使用最少的重复。”a.\*?b”匹配最短的，以a开始，以b结束的字符串。如果把它应用于”aabab”的话，它会匹配aab（第一到第三个字符）和ab（第四到第五个字符）。

|  |  |
| --- | --- |
| 表达式 | 作用 |
| \*? | 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 |

# 在iUAP DI中正则表达式的使用

## 应用正则表达式的转换/作业项

* Excel输入
* 过滤记录
* 数据检验
* 动态列同步
* 批量表同步

等等。

## 应用场景及案例

应用正则表达式，可以实现的功能有：

* **文件名的快速匹配和过滤：案例一**
* **数据过滤规则：案例二和案例三**
* **表名的快速匹配和过滤：案例四**

### 案例一：匹配和过滤文件名

在文本输入和Excel输入中，可以进行同结构多个文件的数据读取，在选文件时可以选择一个文件夹下的文件，通过正则表达式过滤和排除。

示例说明 ：在文本输入中填写正则表达是Linux.\*\.txt,选择C:\Users\zuoym\Documents\目录，点击“增加”按钮，在问价列表中增加了一条记录，点击“显示文件名”，在弹出框内可以看到所有以Linux开头的文本文件。



### 案例二：过滤数据

 在转换的“过滤记录”中可以对数据进行过滤，其中运算符可以选择正则表达式。

 示例说明：如果利用“过滤记录”对省份进行数据过滤，仅仅需要数据流中的以北京，天津或河北的数据，运算符选择REGEXP，值填写(北京|河北|天津).\*，如下图：



### 案例三：数据检验

 在转换的“数据检验”中可以按照正则表达式对数据进行检验。

示例说明：对数据流中的Email和电话进行正则表达式判断，如下截图：





### 案例四：过滤表名

在作业的“动态列同步”和“批量表同步”中可以通过正则表达式选择要同步的表。

示例说明：在动态列同步中，选择ae\_di\_开头的数据表且不区分大小写，如下图：



# 总结

正则表达式是处理字符串的有效手段，iUAP DI中有很多使用正则表达地方，希望通过以上的介绍，使大家能够理解和使用正则表达式并能够使用其在iUAP DI做一些工作。