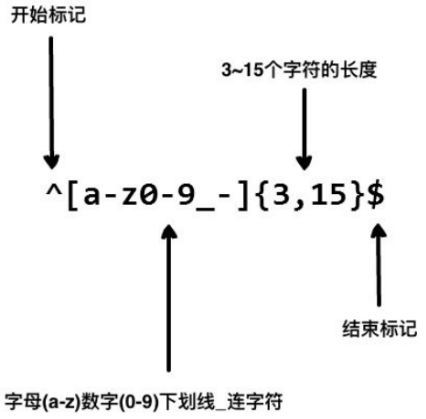
一、什么是正则表达式

2020年4月6日 22:05

正则表达式（ Regular expression）是一组由字母和符号组成的特殊文本，它可以用来从文本中找出满足你想要的格式的句子。

比如我们在网站中看到对用户名规则做出了如下限制：只能包含小写字母、数字、下划线和连字符，并且限制用户名长度在3~15个字符之间，如何验证一个用户名是否符合规则呢？ 我们使用以下正则表达式：



以上的正则表达式可以接受john_doe、jo-hn_doe、john12_as，但不能匹配A1，因为它包含了大写字母而且长度不到3个字符。

1.1 re.compile

2020年4月8日 12:12

compile：re.compile是将正则表达式转换为模式对象，这样可以更有效率匹配。使用compile转换一次之后，以后每次使用模式时就不用进行转换
pattern：写正则表达式
flags：匹配模式

从compile()函数的定义中，可以看出返回的是一个匹配对象，它单独使用就没有任何意义，需要和findall(), search(), match()搭配使用。
compile()与findall()一起使用，返回一个列表。

```
# compile配合findall
import re
a = '0355-67796666'
b = re.compile(r'\d+-\d{8}')
r = re.findall(b,a)
print(r)
```

```
import re
a = '0355-67796666'
b = re.compile(r'\d+-\d{8}')
r = b.findall(a)
print(r)
```

```
# 直接使用findall
import re
a = '0355-67796666'
r = re.findall(r'\d+-\d{8}',a)
print(r)
```

```
# compile配合search
import re
a = '0355-67796666'
正则 = re.compile(r'\d+-\d{8}')
r = re.search(正则,a)
print(r.group())
```

```
# compile配合match
import re
a = '0355-67796666'
正则 = re.compile(r'\d+-\d{8}')
r = re.match(正则,a)
print(r.group())
```

1.2 re.match

2020年4月8日 12:26

```
# match 从字符串的第一个字符开始匹配，如果未匹配到返回None，匹配到则返回一个对象
# match判断正则表达式是否从开始处（首字母）匹配一个字符串，例如第一个不是\d(数字)，返回None
import re
a = 'A83C72D1D8E67'
r = re.match('\d',a)
print(r)                # 返回对象所在位置
print(r.group())    # 返回找到的结果，例如8
print(r.span()) # 返回一个元组表示匹配位置（开始，结束）
```

1.3 re.search

2020年4月8日 12:27

Search与match有些类似，只是搜索整个字符串然后第一个匹配到指定的字符则返回值，未匹配到则返回None。获取值得方法也需要通过group()

从字符串开始往后匹配，一匹配到则返回一个对象。需要通过group来获取匹配到的值。

```
# search 遍历字符串，找到正则表达式匹配的第一个位置
import re
a = 'A83C72D1D8E67'
r = re.search('\d',a)
print(r)
print(r.group())
```

1.4 re.findall

2020年4月8日 12:27

Findall是匹配出字符串中所有跟指定值有关的值，并且以列表的形式返回。未匹配到则返回一个空的列表。匹配出指定字符的所有值，并以列表返回值。

二、简单的模式：字符匹配

2020年4月7日 11:29

元字符	描述
.	句号匹配任意单个字符除了换行符
[]	字符种类，匹配方括号内的任意字符，中括号内每个字符是或(or)的关系
[^]	否定的字符种类，匹配除了方括号里的任意字符
*	匹配0次或无限次，重复在*号之前的字符
+	匹配1次或无限次，重复在+号之前的字符
?	匹配0次或1次，重复在?号之前的字符
{n,m}	匹配num个大括号之前的字符（n<=num<=m)
(xyz)	字符集又称做组，匹配与xyz完全相等的字符串，每个字符是且(and)的关系
|	或运算符，匹配符号前或后的字符
\	转义字符，用于匹配一些保留字符 [ ]、( )、{ }、. 、 * 、 + 、? 、 ^ 、$、 \ 、|
^	从字符串开始位置开始匹配
$	从字符串末端开始匹配

反斜杠后面跟普通字符实现特殊功能

特殊字符	描述
\d	匹配数字，相当于[0-9]
\D	不匹配数字，相当于[^0-9]
\s	匹配空白字符(包括空格、换行符、制表符等)，相当于 [\t\n\r\f\v]
\S	与\s相反，相当于 [^\t\n\r\f\v]
\w	匹配中文，下划线，数字，英文，相当于[a-zA-z0-9_]
\W	与\w相反，匹配特殊字符，如$、&、空格、\n、\t等
\b	匹配单词的开始或结束
\B	与\b相反

2.1 元字符

2020年4月7日 11:36

```
# 提取字符串a中所有的数字，返回结果：['7', '6', '3', '6']
import re
a = '孙悟空7猪八戒6沙和尚3唐僧6白龙马'
r = re.findall('[0-9]',a)
print(r)

# 提取字符串a中所有非数字，返回：['孙', '悟', '空', '猪', '八', '戒', '沙', '和', '尚', '唐', '僧', '白', '龙', '马']
import re
a = '孙悟空7猪八戒6沙和尚3唐僧6白龙马'
r = re.findall('[^0-9]',a)
print(r)

# 找到字符串中间字母是d或e的单词，返回：['xdz', 'xez']
import re
a = 'xyz,xcz,xfz,xdz,xaz,xez'
r = re.findall('x[de]z',a)
print(r)

# 找到字符串中间字母不是d或e的单词，返回：['xyz', 'xcz', 'xfz', 'xaz']
import re
a = 'xyz,xcz,xfz,xdz,xaz,xez'
r = re.findall('x[^de]z',a)
print(r)

# 找到字符串中间字母是d或e或f的单词，返回：['xfz', 'xdz', 'xez']
import re
a = 'xyz,xcz,xfz,xdz,xaz,xez'
r = re.findall('x[d-f]z',a)
print(r)
```

2.2 概括字符集

2020年4月7日　　11:57

```
# 提取字符串a中所有的数字
import re
a = 'Excel 12345Word\n23456_PPT12lr'
r = re.findall('\d',a)
print(r)

# 提取字符串a中所有非数字
import re
a = 'Excel 12345Word\n23456_PPT12lr'
r = re.findall('\D',a)
print(r)

# \w 可以提取中文，英文，数字和下划线，不能提取特殊字符
import re
a = 'Excel 12345Word\n23456_PPT12lr'
r = re.findall('\w',a)
print(r)

# \W 提取特殊字符、空格、\n、\t等
import re
a = 'Excel 12345Word\n23456_PPT12lr'
r = re.findall('\W',a)
print(r)
```

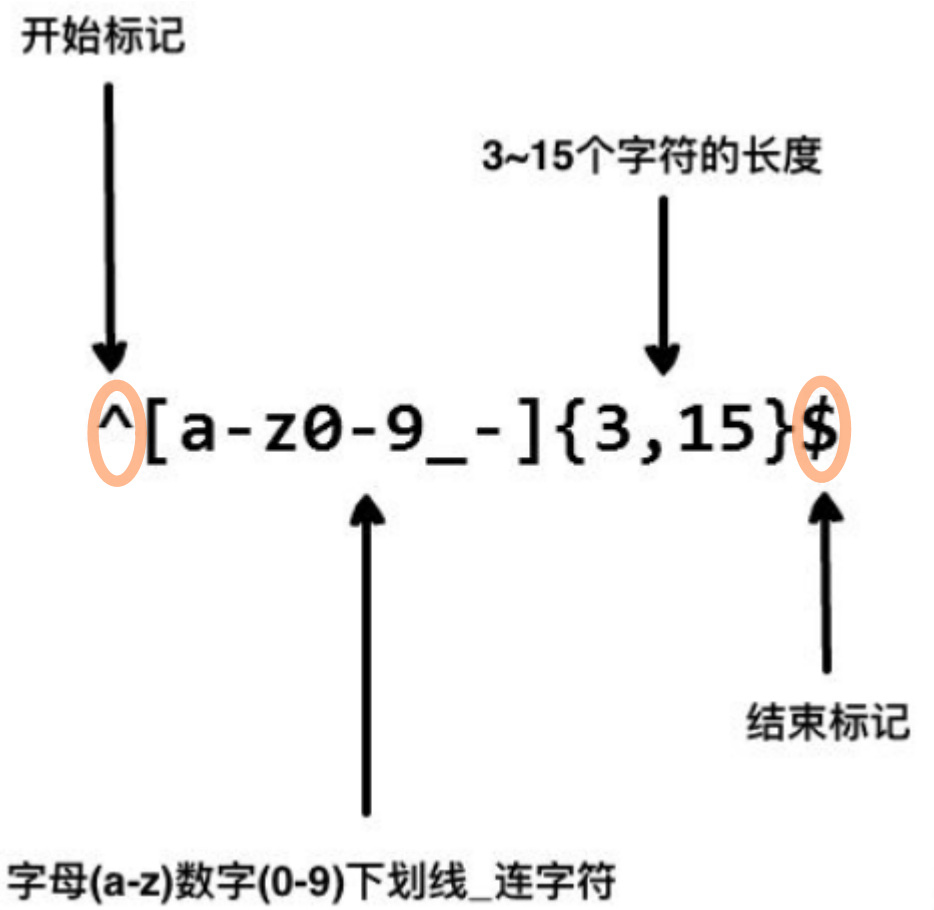
2.3 数量词

2020年4月7日 12:48

```
# 提取大小写字母混合的单词
import re
a = 'Excel 12345Word23456PPT12Lr'
r = re.findall('[a-zA-Z]{3,5}',a)
print(r)

# 贪婪 与 非贪婪 【Python默认使用贪婪模式】
贪婪：'[a-zA-Z]{3,5}'
 # 先找三个连续的字母，最多找到5个连续的字母后停止。在3个以后且5个以内发现了不是子母的也停止。
非贪婪：'[a-zA-Z]{3,5}?' 或 '[a-zA-Z]{3}'
建议使用后者，不要使用?号，否则你会与下面的?号混淆

# 匹配0次或无限多次 *号，*号前面的字符出现0次或无限次
import re
a = 'exce0excell3excel3'
r = re.findall('excel*',a)
print(r)

# 匹配1次或者无限多次 +号，+号前面的字符至少出现1次
import re
a = 'exce0excell3excel3'
r = re.findall('excel+',a)
print(r)

# 匹配0次或1次 ?号，?号经常用来去重复
import re
a = 'exce0excell3excel3'
r = re.findall('excel?',a)
print(r)
```

2.4 边界匹配 ^和$

2020年4月7日 14:12




```
# 限制电话号码的位置必需是8-11位才能提取
import re
tel = '13811115888'
r = re.findall('^\d{8,11}$',tel)
print(r)
```

2.5 组 ( )

2020年4月7日 14:27

```
# 将abc打成一个组，{2}指的是重复几次，匹配abcabc
import re
a = 'abcabcabcxyzabcabcxyzabc'
r = re.findall('(abc){2}',a)
print(r)
```

可以加入很多个组

2.6 匹配模式参数

2020年4月7日 21:02

```
# findall第三参数 re.I忽略大小写
import re
a = 'abcFBIabcCIAabc'
r = re.findall('fbi',a,re.I)
print(r)

# 多个模式之间用 | 连接在一起
import re
a = 'abcFBI\nabcCIAabc'
r = re.findall('fbi.{1}',a,re.I | re.S)  # 匹配fbi然后匹配任意一个字符包括\n
print(r)
```

注：.句号，不匹配\n，但是使用re.S之后，匹配所有字符包括换行符

1).re.I(re.IGNORECASE): 忽略大小写

2).re.M(MULTILINE): 多行模式，改变’ ^’ 和’ $’ 的行为

3).re.S(DOTALL): 点任意匹配模式，改变’.’ 的行为

4).re.L(LOCALE): 使预定字符类 \w \W \b \B \s \S 取决于当前区域设定

5).re.U(UNICODE): 使预定字符类 \w \W \b \B \s \S \d \D 取决于unicode定义的字符属性

6).re.X(VERBOSE): 详细模式。这个模式下正则表达式可以是多行，忽略空白字符，并可以加入注释

2.7 re.sub替换字符串

2020年4月7日 21:18

```
# 把FBI替换成BBQ
import re
a = 'abcFBIabcCIAabc'
r = re.sub('FBI','BBQ',a)
print(r)

# 把FBI替换成BBQ，第4参数写1，证明只替换第一次，默认是0（无限替换）
import re
a = 'abcFBIabcFBIaFBICIAabc'
r = re.sub('FBI','BBQ',a,1)
print(r)
```

注意：虽然字符串的内置方法 字符串.replace 也可以进行替换，但是正则表达式更强大

```
# 拓展知识
import re
a = 'abcFBIabcFBIaFBICIAabc'
def 函数名(形参):
    pass
r = re.sub('FBI',函数名,a,1)
print(r)
```

分析：如果找到了FBI这个子串，会将FBI当成参数传到形参中，pass是什么都没返回，所以FBI被空字符串代替了。

```
# 把函数当参数传到sub的列表里，实现把业务交给函数去处理，例如将FBI替换成$FBI$
import re
a = 'abcFBIabcFBIaFBICIAabc'
def 函数名(形参):
    分段获取 = 形参.group()          # group（）在正则表达式中用于获取分段截获的字符串，获取到FBI
    return '$' + 分段获取 + '$'
r = re.sub('FBI',函数名,a)
print(r)
```

2.8 把函数做为参数传递

2020年4月7日　　21:54

```
# 将字符串中大于等于5的替换成9，小于5的替换成0
import re
a = 'C52730A52730D52730'
def 函数名(形参):
    分段获取 = 形参.group()
    if int(分段获取) >= 5:
        return '9'
    else:
        return '0'
r = re.sub('\d',函数名,a)
print(r)
```

2.9 group分组

2020年4月7日 22:38

```
import re
a = "123abc456"
print re.search("([0-9]*)([a-z]*)([0-9]*)",a).group(0)   #123abc456,返回整体
print re.search("([0-9]*)([a-z]*)([0-9]*)",a).group(1)   #123
print re.search("([0-9]*)([a-z]*)([0-9]*)",a).group(2)   #abc
print re.search("([0-9]*)([a-z]*)([0-9]*)",a).group(3)   #456
```

1. 正则表达式中的三组括号把匹配结果分成三组
group() 同group（0）就是匹配正则表达式整体结果
group(1) 列出第一个括号匹配部分，group(2) 列出第二个括号匹配部分，group(3) 列出第三个括号匹配部分。
2. 没有匹配成功的，re.search（）返回None
3. 当然正则表达式中没有括号，group(1)肯定不对了。

```
import re
a = 'life is short,i use python'
r = re.search('life(.*)python',a)
print(r.group(1))
```

等同于

```
import re
a = 'life is short,i use python'
r = re.findall('life(.*)python',a)
print(r)
```

```
# 拓展知识
import re
a = 'life is short,i use python,i love python'
r = re.search('life(.*)python(.*)python',a)
print(r.group(0))       # 完整正则匹配
print(r.group(1))       # 第1个分组之间的取值
print(r.group(2))       # 第2个分组之间的取值
print(r.group(0,1,2)) # 以元组形式返回3个结果取值
print(r.groups())       # 返回就是group(1)和group(2)
```

三、正则表达式的一些建议

2020年4月7日 22:53

1、常用的正则表达式，不用自己写，在百度上搜索常用正则表达式

2、如果内置方法可以快速解决问题，建议不要化简为繁

附1：正则表达式基础语法

2020年4月8日 10:48

^	指出一个字符串的开始
$	指出一个字符串的结束
\	将下一个字符标记为一个特殊字符、或一个原义字符、或一个向后引用、或一个八进制转义符
^abc	匹配所有以 "abc" 开始的字符串 （例如："abc"，"abccba"）
abc$	匹配所有以"abc" 结尾的字符串 （例如："gggabc"，"reddcba"）
^abc$	匹配开始和结尾都为"abc"的字符串 （例如："abc"）
abc	没有任何字符，匹配任何包含"abc"的字符串 （例如："aaaabccc"，"abc123"）
n	匹配n "\n"：匹配换行符 "\/"这里是 \ he / 连在一起写，匹配 " / " 字符

*	匹配前面的子表达式零次或多次
+	匹配前面的子表达式一次或多次
?	匹配前面的子表达式零次或一次
ac*	匹配字符串其中一个a后面跟着零个或若干个c （例如："accc"，"abbb"）
ac+	匹配字符串其中一个a后面跟着至少一个c或者多个 （例如："ac"，"acccccccc"）
ac?	匹配字符串其中一个a后面跟着零个或者一个c （例如："a"，"ac"）
a?c+$	匹配字符串的末尾有零个或一个a跟着一个或多个c （例如："ac"，"acccccc"，''c''，"ccccccc"）

{n}	n为非负整数，匹配n次
{n，}	n为非负整数，匹配至少n次
{n，m}	n，m为非负整数，最少匹配n次 最多匹配m次
ab{3}	表示一个字符串有一个a后面跟随2个b （例如："abb"，"abbbbb"）
ab{3,}	表示一个字符串有一个a后面跟随至少2个b （例如："abb"，"abbb"）
ab{3,6}	表示一个字符串有一个a后面跟随3到6个b （例如："abbb"，"abbbb"，"abbbb"）

|	表示"或"
.	表示任何字符
a|b	表示一个字符串里有 a 或者 b （例如："a"，"b"，"ab"，"abc"）
a.	表示一个字符串有一个 a 后面跟着一个任意字符 （例如："a1"，"a456"，"avv"）

附2：方括号里用"^"表示不希望出现的字符

2020年4月8日 11:15

[abc]	**表示字符集合，表示一个字符串有一个"a"或"b"或"c" 等价于 [z|b|c]**
[^abc]	**表示一个字符串中不应该出现abc，即是匹配未包含改集合的任意字符**
[a-z]	**表示一个字符串中存在一个a和z之间的所有字母**
[0-9]	**表示一个字符串中存在一个0和9之间的所有数字**
[^a-z]	**表示一个字符串中不应该出现a到z之间的任意一个字母**
^[a-zA-Z]	**表示一个字符串中以字母开头**
[0-9]%	**表示一个百分号前有一个的数字；**

附3：由字符'\'和另一个字符组成特殊含义

2020年4月8日 11:17

\d	匹配一个数字字符，等价[0-9]
\D	匹配一个非数字字符，等价[^0-9]
\f	匹配一个换页符，等价\x0c和\cL
\n	匹配一个换行符。等价于\x0a和\cJ
\r	匹配一个回车符。等价于\x0d和\cM
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于[\f\n\r\t\v]
\S	匹配任何非空白字符。等价于[^ \f\n\r\t\v]
\t	匹配一个制表符。等价于\x09和\cI
\v	匹配一个垂直制表符。等价于\x0b和\cK
\w	匹配包括下划线的任何单词字符。等价于“[A-Za-z0-9_]”
\W	匹配任何非单词字符。等价于“[^A-Za-z0-9_]”

附4：正则表达式实战

2020年4月8日 11:22

/^[0-9]{1,20}$/

表示字符串全部由数字组成，即是匹配当前字符串是否是由全数字组成

[0-9] 表示字符的范围是0到9

{1,20}表示字符串长度至少为1，最多为20，字符串出现次数的范围

/^[a-zA-Z]{1}([a-zA-Z0-9._]){5,15}$/

可以用来验证登录名，首字母为字母，长度为至少6最多16

^[a-zA-Z]{1} 表示最开始的第一个首字母为字母

([a-zA-Z0-9._]){5,15} 这是首字母后面的即是从第二个字母开始要求至少再有5个最多15个由字母数字以及指定特殊字符组成的字符串

/^[1][3|4|5|8][0-9]\d{8}$/

可以用来验证手机号码，首字母为1，长度11，首尾都是数字

^[1] 第一个数字为1

[3|4|5|8] 第二个数字为 3或者4或者5或者8

[0-9]\d{8} 匹配一个数字范围是0-9，匹配8次，所以至少要有8个数字。加起来就是11个

/^(\w){6,20}$/

验证密码

\w 匹配任何非单词字符 等价于“[^A-Za-z0-9_]”

(\w){6,20} 匹配任何非单词字符，最少6个最多20个

附5：RegExp 对象的属性和方法

2020年4月8日 11:25

下面我们来研究如何使用

//构造一个正则对象，并填写表达式

```
var re = new RegExp("\[0-9]*");
var re = /[0-9]*/;
```

//注意：当使用构造函数（new RegExp）创造正则对象时，需要常规的字符转义规则（在前面加反斜杠 \）

test() 方法：正则表达式方法。

test() 方法用于检测一个字符串是否匹配某个模式，如果字符串中含有匹配的文本，则返回 true，否则返回 false。

```
var re = /[0-9]*/;

re.test("abc") //返回false

re.test("1234") //返回true
```

exec() 方法：一个正则表达式方法。

exec() 方法用于检索字符串中的正则表达式的匹配。

该函数返回一个数组，其中存放匹配的结果。如果未找到匹配，则返回值为 null。

```
var re = /[0-9]*/;

re.exec("abc") //返回 null

re.exec("1234") //返回1234
```

附6：常用正则表达式

2020年4月8日 11:26

附6.1 校验数字的表达式

2020年4月8日 11:27

数字	^[0-9]$
n位的数字	^\d{n}$
至少n位的数字	^\d{n,}$
m-n位的数字	^\d{m,n}$
零和非零开头的数字	^(0|[1-9][0-9]*)$
非零开头的最多带两位小数的数字	^([1-9][0-9]*)+(.[0-9]{1,2})?$
带1-2位小数的正数或负数	^(\-)?\d+(\.\d{1,2})?$
正数、负数、和小数	^(\-|\+)?\d+(\.\d+)?$
有两位小数的正实数	^[0-9]+(.[0-9]{2})?$
有1~3位小数的正实数	^[0-9]+(.[0-9]{1,3})?$
非零的正整数	^[1-9]\d*$ 或 ^([1-9][0-9]*){1,3}$ 或 ^\+?[1-9][0-9]*$
非零的负整数	^\-[1-9][]0-9"*$ 或 ^-[1-9]\d*$
非负整数	^\d+$ 或 ^[1-9]\d*|0$
非正整数	^-[1-9]\d*|0$ 或 ^((-\d+)|(0+))$
非负浮点数	^\d+(\.\d+)?$ 或 ^[1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.0+|0$
非正浮点数	^((-\d+(\.\d+)?)|(0+(\.0+)?))$ 或 ^(-([1-9]\d*\.\d*|0\.\d*[1-9]\d*))|0?\.0+|0$
正浮点数	^[1-9]\d*\.\d*|0\.\d*[1-9]\d*$ 或 ^(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*))$
负浮点数	^-([1-9]\d*\.\d*|0\.\d*[1-9]\d*)$ 或 ^(-(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*)))$
浮点数	^(-?\d+)(\.\d+)?$ 或 ^-?([1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.0+|0)$

附6.2 校验字符的表达式

2020年4月8日 11:31

汉字	^[\u4e00-\u9fa5]{0,}$
英文和数字	^[A-Za-z0-9]+$ 或 ^[A-Za-z0-9]{4,40}$
长度为3-20的所有字符	^.{3,20}$
由26个英文字母组成的字符串	^[A-Za-z]+$
由26个大写英文字母组成的字符串	^[A-Z]+$
由26个小写英文字母组成的字符串	^[a-z]+$
由数字和26个英文字母组成的字符串	^[A-Za-z0-9]+$
由数字、26个英文字母或者下划线组成的字符串	^\w+$ 或 ^\w{3,20}$
中文、英文、数字包括下划线	^[\u4E00-\u9FA5A-Za-z0-9_]+$
中文、英文、数字但不包括下划线等符号	^[\u4E00-\u9FA5A-Za-z0-9]+$ 或 ^[\u4E00-\u9FA5A-Za-z0-9]{2,20}$
可以输入含有^%&',;=?$\"等字符	[^%&',;=?$\x22]+
禁止输入含有~的字符	[^~\x22]+

附6.3 特殊需求表达式1

2020年4月8日 11:34

Email地址	^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$
域名	[a-zA-Z0-9][-a-zA-Z0-9]{0,62}(/.[a-zA-Z0-9][-a-zA-Z0-9]{0,62})+/.?
InternetURL	[a-zA-z]+://[^\s]* 或 ^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]*)?$
手机号码	^(13[0-9]|14[5|7]|15[0|1|2|3|5|6|7|8|9]|18[0|1|2|3|5|6|7|8|9])\d{8}$
电话号码	("XXX-XXXXXXX"、"XXXX-XXXXXXXX"、"XXX-XXXXXXX"、"XXX-XXXXXXXX"、"XXXXXXX"和"XXXXXXXX")：^(\(\d{3,4}-)|\d{3.4}-)?\d{7,8}$
国内电话号码(0511-4405222、021-87888822)	\d{3}-\d{8}|\d{4}-\d{7}
身份证号(15位、18位数字)	^\d{15}|\d{18}$
短身份证号码(数字、字母x结尾)	^([0-9]){7,18}(x|X)?$ 或 ^\d{8,18}|[0-9x]{8,18}|[0-9X]{8,18}?$
帐号是否合法(字母开头，允许5-16字节，允许字母数字下划线)	^[a-zA-Z][a-zA-Z0-9_]{4,15}$
密码(以字母开头，长度在6~18之间，只能包含字母、数字和下划线)	^[a-zA-Z]\w{5,17}$
强密码(必须包含大小写字母和数字的组合，不能使用特殊字符，长度在8-10之间)	^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,10}$
日期格式	^\d{4}-\d{1,2}-\d{1,2}
一年的12个月(01～09和1～12)	^(0?[1-9]|1[0-2])$
一个月的31天(01～09和1～31)	^((0?[1-9])|((1|2)[0-9])|30|31)$
有四种钱的表示形式我们可以接受:"10000.00" 和 "10,000.00", 和没有 "分" 的 "10000" 和 "10,000"	^[1-9][0-9]*$
这表示任意一个不以0开头的数字,但是,这也意味着一个字符"0"不通过,所以我们采用下面的形式	^(0|[1-9][0-9]*)$
一个0或者一个不以0开头的数字.我们还可以允许开头有一个负号	^(0|-?[1-9][0-9]*)$
这表示一个0或者一个可能为负的开头不为0的数字.让用户以0开头好了.把负号的也去掉,因为钱总不能是负的吧.下面我们要加的是说明可能的小数部分	^[0-9]+(.[0-9]+)?$
必须说明的是,小数点后面至少应该有1位数,所以"10."是不通过的,但是 "10" 和 "10.2" 是通过的	^[0-9]+(.[0-9]{2})?$
这样我们规定小数点后面必须有两位,如果你认为太苛刻了,可以这样	^[0-9]+(.[0-9]{1,2})?$
这样就允许用户只写一位小数.下面我们该考虑数字中的逗号了,我们可以这样	^[0-9]{1,3}(,[0-9]{3})*(.[0-9]{1,2})?$
1到3个数字,后面跟着任意个 逗号+3个数字,逗号成为可选,而不是必须	^([0-9]+|[0-9]{1,3}(,[0-9]{3})*)(.[0-9]{1,2})?$
备注：这就是最终结果了,别忘了"+"可以用"*"替代如果你觉得空字符串也可以接受的话(奇怪,为什么?)	最后,别忘了在用函数时去掉去掉那个反斜杠,一般的错误都在这里
xml文件	^([a-zA-Z]+-?)+[a-zA-Z0-9]+\\.[x|X][m|M][l|L]$
中文字符的正则表达式	[\u4e00-\u9fa5]
双字节字符	[^\x00-\xff] (包括汉字在内，可以用来计算字符串的长度(一个双字节字符长度计2，ASCII字符计1))28 空白行的正则表达式：\n\s*\r (可以用来删除空白行)
HTML标记的正则表达式	<(\S*?)[^>]*>.*?</\1>|<.*? /> (网上流传的版本太糟糕，上面这个也仅仅能部分，对于复杂的嵌套标记依旧无能为力)30 首尾空白字符的正则表达式：^\s*|\s*$或(^\s*)\|(\s*$) (可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等等)，非常有用的表达式)
腾讯QQ号	[1-9][0-9]{4,} (腾讯QQ号从10000开始)
中国邮政编码	[1-9]\d{5}(?!\d) (中国邮政编码为6位数字)
IP地址(提取IP地址时有用)	\d+\.\d+\.\d+\.\d+
IP地址	((?:(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d)\\.){3}(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d))

附6.4 特殊需求表达式2

2020年4月8日 11:45

"^\d+$" //非负整数（正整数 + 0）

"^[0-9]*[1-9][0-9]*$" //正整数

"^((-\d+)|(0+))$" //非正整数（负整数 + 0）

"^-[0-9]*[1-9][0-9]*$" //负整数

"^-?\d+$" //整数

"^\d+(\.\d+)?$" //非负浮点数（正浮点数 + 0）

"^(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*))$" //正浮点数

"^((-\d+(\.\d+)?)|(0+(\.0+)?))$" //非正浮点数（负浮点数 + 0）

"^(-(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*)))$" //负浮点数

"^(-?\d+)(\.\d+)?$" //浮点数

"^[A-Za-z]+$" //由26个英文字母组成的字符串

"^[A-Z]+$" //由26个英文字母的大写组成的字符串

"^[a-z]+$" //由26个英文字母的小写组成的字符串

"^[A-Za-z0-9]+$" //由数字和26个英文字母组成的字符串

"^\w+$" //由数字、26个英文字母或者下划线组成的字符串

"^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$" //email地址

"^[a-zA-z]+://(\w+(-\w+)*)(\.(\w+(-\w+)*))*(\?\S*)?$" //url

整数或者小数：^[0-9]+\.{0,1}[0-9]{0,2}$

只能输入数字："^[0-9]*$"。

只能输入n位的数字："^\d{n}$"。

只能输入至少n位的数字："^\d{n,}$"。

只能输入m~n位的数字：。"^\d{m,n}$"

只能输入零和非零开头的数字："^(0|[1-9][0-9]*)$"。

只能输入有两位小数的正实数："^[0-9]+(.[0-9]{2})?$"。

只能输入有1~3位小数的正实数："^[0-9]+(.[0-9]{1,3})?$"。

只能输入非零的正整数："^\+?[1-9][0-9]*$"。

只能输入非零的负整数："^\-[1-9][]0-9"*$。

只能输入长度为3的字符："^.{3}$"。

只能输入由26个英文字母组成的字符串："^[A-Za-z]+$"。

只能输入由26个大写英文字母组成的字符串："^[A-Z]+$"。

只能输入由26个小写英文字母组成的字符串："^[a-z]+$"。

只能输入由数字和26个英文字母组成的字符串："^[A-Za-z0-9]+$"。

只能输入由数字、26个英文字母或者下划线组成的字符串："^\w+$"。

验证用户密码："^[a-zA-Z]\w{5,17}$"正确格式为：以字母开头，长度在6~18之间，只能包含字符、数字和下划线。

验证是否含有^%&',;=?$\"等字符："[^%&',;=?$\x22]+"。

只能输入汉字："^[\u4e00-\u9fa5]{0,}$"

验证Email地址："^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$"。

验证InternetURL："^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]*)?$"。

验证电话号码："^(\(\d{3,4}-)|\d{3.4}-)?\d{7,8}$"正确格式为："XXX-XXXXXXX"、"XXXX-XXXXXXXX"、"XXX-XXXXXXX"、"XXX-XXXXXXXX"、"XXXXXXX"和"XXXXXXXX"。

验证身份证号（15位或18位数字）："^\d{15}|\d{18}$"。

验证一年的12个月："^(0?[1-9]|1[0-2])$"正确格式为："01"～"09"和"1"～"12"。

验证一个月的31天："^((0?[1-9])|((1|2)[0-9])|30|31)$"正确格式

为；"01"～"09"和"1"～"31"。

整数或者小数：^[0-9]+\.{0,1}[0-9]{0,2}$

"^\w+$" //由数字、26个英文字母或者下划线组成的字符串

"^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$" //email地址

"^[a-zA-z]+://(\w+(-\w+)*)(\.(\w+(-\w+)*))*(\?\S*)?$" //url

可输入形如2008、2008-9、2008-09、2008-9-9、2008-09-09.　^(\d{4}|(\d{4}-\d{1,2})|(\d{4}-\d{1,2}-\d{1,2}))$

邮箱验证正则表达式　\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*

附6.5 网络验证应用技巧

2020年4月8日 11:46

验证 E-mail格式

```
public bool IsEmail(string str_Email) { return System.Text.RegularExpressions.Regex.IsMatch(str_Email,@"^([\w-\.]+)@(([—9]1,3\.[—9]1,3\.[—9]1,3\.)|(([\w—]+\.)+))([a—zA—Z]2,4|[—9]1,3)(?)$"); }
```

验证 IP 地址

```
public bool IPCheck(string IP) {string num = "(25[0-5]|2[0-4]\\d|[0-1]\\d{2}|[1-9]?\\d)"; return Regex.IsMatch(IP,("^" + num + "\\." + num + "\\." + num + "\\." + num + "$")); }
```

验证 URL

```
public bool IsUrl(string str_url) { return System.Text.RegularExpressions.Regex.IsMatch(str_url, @"http(s)?://([\w-]+\.)+[\w-]+(/[\w- ./?%&=]*)?"); }
```

附6.6 常用数字验证技巧

2020年4月8日 11:49

验证电话号码

```
public bool IsTelephone(string str_telephone) { return System.Text.RegularExpressions.Regex.IsMatch(str_telephone, @"^(\d{3,4}-)?\d{6,8}$"); }
```

输入密码条件(字符与数据同时出现)

```
public bool IsPassword(string str_password) { return System.Text.RegularExpressions.Regex.IsMatch(str_password, @"[A-Za-z]+[0-9]"); }
```

邮政编号

```
public bool IsPostalcode(string str_postalcode) { return System.Text.RegularExpressions.Regex.IsMatch(str_postalcode, @"^\d{6}$"); }
```

手机号码

```
public bool IsHandset(string str_handset) { return System.Text.RegularExpressions.Regex.IsMatch(str_handset, @"^[1]+[3,5]+\d{9}$"); }
```

身份证号

```
public bool IsIDcard(string str_idcard) { return System.Text.RegularExpressions.Regex.IsMatch(str_idcard, @"(^\d{18}$)|(^\d{15}$)"); }
```

两位小数

```
public bool IsDecimal(string str_decimal) { return System.Text.RegularExpressions.Regex.IsMatch(str_decimal, @"^[0-9]+(.[0-9]{2})?$"); }
```

一年的12个月

```
public bool IsMonth(string str_Month) { return System.Text.RegularExpressions.Regex.IsMatch(str_Month, @"^(0?[[1-9]|1[0-2])$"); }
```

一个月的31天

```
public bool IsDay(string str_day) { return System.Text.RegularExpressions.Regex.IsMatch(str_day, @"^((0?[1-9])|((1|2)[0-9])|30|31)$"); }
```

数字输入

```
public bool IsNumber(string str_number) { return System.Text.RegularExpressions.Regex.IsMatch(str_number, @"^[0-9]*$"); }
```

密码长度 (6-18位)

```
public bool IsPasswLength(string str_Length) { return System.Text.RegularExpressions.Regex.IsMatch(str_Length, @"^\d{6,18}$"); }
```

非零的正整数

```
public bool IsIntNumber(string str_intNumber) { return System.Text.RegularExpressions.Regex.IsMatch(str_intNumber, @"^\+?[1-9][0-9]*$"); }
```

附6.7 常用字符验证技巧

2020年4月8日 11:50

大写字母

```
public bool IsUpChar(string str_UpChar) { return System.Text.RegularExpressions.Regex.IsMatch(str_UpChar, @"^[A-Z]+$"); }
```

小写字母

```
public bool IsLowChar(string str_UpChar) { return System.Text.RegularExpressions.Regex.IsMatch(str_UpChar, @"^[a-z]+$"); }
```

检查字符串重复出现的词

```
private void btnWord_Click(object sender, EventArgs e) { System.Text.RegularExpressions.MatchCollection matches =
System.Text.RegularExpressions.Regex.Matches(label1.Text, @"\b(?<word>\w+)\s+(\k<word>)\b",
System.Text.RegularExpressions.RegexOptions.Compiled | System.Text.RegularExpressions.RegexOptions.IgnoreCase); if (matches.Count != 0) {
foreach (System.Text.RegularExpressions.Match match in matches) {
string word = match.Groups["word"].Value; MessageBox.Show(word.ToString(),"英文单词"); } } else { MessageBox.Show("没有重复的单词"); } }
```

替换字符串

```
private void button1_Click(object sender, EventArgs e) {
string strResult = System.Text.RegularExpressions.Regex.Replace(textBox1.Text, @"[A-Za-z]\*?", textBox2.Text); MessageBox.Show("替换前字符:" +
"\n" + textBox1.Text + "\n" + "替换的字符:" + "\n" + textBox2.Text + "\n" + "替换后的字符:" + "\n" + strResult,"替换"); }
```

拆分字符串

```
private void button1_Click(object sender, EventArgs e) { //实例: 甲025-8343243乙0755-2228382丙029-32983298389289328932893289丁
foreach (string s in System.Text.RegularExpressions.Regex.Split(textBox1.Text,@"\d{3,4}-\d*")) {
textBox2.Text+=s; //依次输出 "甲乙丙丁" } }
```

验证输入字母

```
public bool IsLetter(string str_Letter) { return System.Text.RegularExpressions.Regex.IsMatch(str_Letter, @"^[A-Za-z]+$"); }
```

验证输入汉字

```
public bool IsChinese(string str_chinese) { return System.Text.RegularExpressions.Regex.IsMatch(str_chinese, @"^[\u4e00-\u9fa5],{0,}$"); }
```

验证输入字符串 (至少8个字符)

```
public bool IsLength(string str_Length) { return System.Text.RegularExpressions.Regex.IsMatch(str_Length, @"^.{8,}$"); }
```