正则表达式入门:正则表达式 学习参考 推荐 入门者看

疯狂代码 http://CrazyCoder.cn/ ĵ:http:/CrazyCoder.cn/DeveloperUtil/Article69995.html

1 概述正则表达式(Regular Expression)是种匹配模式描述是串文本特征

正如自然语言中“高大”、“坚固”等词语抽象出来描述事物特征样正则表达式就是高度抽象用来描述串特征

正则表达式(以下简称正则Regex)通常不独立存在各种编程语言和工具作为宿主语言提供对正则支持并根据自身语言特点进行定剪裁或扩展

正则入门很容易有限语法规则很容易掌握但是目前正则普及率并不高主要是正则流派众多各种宿主语言提供文档都过多关注于自身些细节而这些细节通常是初学者并不需要关注

当然如果想要深入了解正则表达式这些细节又是必须被关注这是后话让我们先从正则基础开始进入正则表达式世界

2 正则表达式基础2.1 基本概念2.1.1 串组成

对于串“a5”是由两个“a”、“5”以及 3个位置组成这点对于正则表达式匹配原理理解很重要

2.1.2 占有和零宽度正则表达式匹配过程中如果子表达式匹配到是内容而非位置并被保存到最终匹配结果中那么就认为这个子表达式是占有；如果子表达式匹配仅仅是位置或者匹配内容并不保存到最终匹配结果中那么就认为这个子表达式是零宽度

占有还是零宽度是针对匹配内容是否保存到最终匹配结果中而言

占有是互斥零宽度是非互斥也就是个同时间只能由个子表达式匹配而个位置却可以同时由多个零宽度子表达式匹配

2.1.3 正则表达式构成正则表达式由两种构成种是在正则表达式中具体特殊意义“元”另种是普通“文本”

元可以是个如“^”也可以是个序列如“\w”

2.2 元(Meta Character)2.2.1 […] 组(Character Classes) 组可以匹配[ ]中包含任意个虽然可以是任意个但只能是个

组支持由连“-”来表示个范围当“-”前后构成范围时要求前面码位小于后面码位

[^...] 排除型组排除型组表示任意个未列出同样只能是个排除型组同样支持由连“-”来表示个范围

表达式

介绍说明

[abc]

表示“a”或“b”或“c”

[0-9]

表示0~9中任意个数字等价于[0123456789]

[\u4e00-\u9fa5]

表示任意个汉字

[^a1<]

表示除“a”、“1”、“<”外其它任意个

[^a-z]

表示除小写字母外任意个

举例:

“[0-9][0-9]”在匹配“Windows 2003”时匹配成功匹配结果为“20”

“[^inW]”在匹配“Windows 2003”时匹配成功匹配结果为“d”

2.2.2 常见范围缩写对于些常用范围如数字等由于非常常用即使使用[0-9]这样组仍显得麻烦所以定义了些元来表示常见范围

表达式	介绍说明
\d	任意个数字相当于[0-9]即0~9 中任意个
\w	任意个字母或数字或下划线相当于[a-zA-Z0-9_]
\s	任意空白相当于[\r\n\f\t\v]
\D	

任意个非数字\d取反相当于[^0-9]

\W

\w取反相当于[^a-zA-Z0-9_]

\S

任意非空白\s取反相当于[^ \r\n\f\t\v]

举例:

“\w\s\d”在匹配“Windows 2003”时匹配成功匹配结果为“s 2”

2.2.3 . 小数点小数点可以匹配除“\n”以外任意个如果要匹配包括“\n”在内所有般用[\s\S]或者是用“.”加(?s)匹配模式来实现

表达式

介绍说明

.

匹配除了换行符 \n 以外任意个

2.2.4 其它元

表达式

介绍说明

^

匹配串开始位置不匹配任何

$

匹配串结束位置不匹配任何

\b

匹配单词边界不匹配任何

举例:

“^a”在匹配“cba”时匹配失败表达式要求开始位置后面是“a”而“cba”显然是不满足

“\d$”在匹配“123”时匹配成功匹配结果为“3”这个表达式要求匹配结尾处数字如果结尾处不是数字如“123abc”则是匹配失败

2.2.5 转义些不可见或是在正则中具有特殊意义元如想匹配本身需要用“\”对其进行转义

表达式	介绍说明
\r\n	回车和换行
\\	匹配“\”本身
\^\$\.	分别匹配“^”、“$”和“.”

以下在匹配其本身时通常需要进行转义在实际应用中根据具体情况需要转义可能不止如下所列

. $ ^ { [(|) * + ? \

2.2.6 量词(Quantier)量词表示个子表达式可以匹配次数量词可以用来修饰个、组或是用括起来子表达式些常用量词被定义成独立元

表达式

介绍说明

举例

{m}

表达式匹配m次

“\d{3}”相当于“\d\d\d ”

“(abc){2}”相当于“abcabc”

{m,n}

表达式匹配最少m次最多n次

“\d{2,3}”可以匹配“12”或“321”等2到3位数字

{m,}

表达式至少匹配m次

“[a-z]{8,}”表示至少8位以上字母

?

表达式匹配0次或1次相当于{0,1}

“ab?”可以匹配“a”或“ab”

*

表达式匹配0次或任意多次相当于{0,}

“<[^>]*>”中“[^>]*”表示0个或任意多个不是“>”

+

表达式匹配1次或意多次至少1次相当于{1,}

“\d\s+\d”表示两个数字中间至少有个以上空白

注意:在不是动态生成正则表达式中不要出现“{1}”这样量词如“\w{1}”在结果上等价于“\w”但是会降低匹配效率和可读性属于画蛇添足做法

2.2.7 分支结构(Alternation)当个串某子串具有多种可能时采用分支结构来匹配“|”表示多个子表达式的间“或”关系“|”是以限定范围如果在“|”左右两侧没有来限定范围那么它作用范围即为“|”左右两侧整体

表达式

介绍说明

|

多个子表达式的间取“或”关系

举例:

“^aa|b$”在匹配“cccb”时是可以匹配成功匹配结果是“b”这个表达式表示匹配“^aa”或“b$”而“b$”在匹配“cccb ”时是可以匹配成功

“^(aa|b)$”在区配“cccb”时是匹配失败这个表达式表示在“开始”和“结束”位置的间只能是“aa”或“b”而“cccb”显然是不满足

3　正则表达式进阶3.1　捕获组(Capture Group)捕获组就是把正则表达式中子表达式匹配内容保存到内存中以数字编号或手动命名组里以供后面引用

表达式	介绍说明
(Expression)	普通捕获组将子表达式Expression匹配内容保存到以数字编号组里
(?<name> Expression)	命名捕获组将子表达式Expression匹配内容保存到以name命名组里

普通捕获组(在不产生歧义情况下简称捕获组)是以数字进行编号编号规则是以“(”从左到右出现顺序从1开始进行编号通常情况下编号为0组表示整个表达式匹配内容

命名捕获组可以通过捕获组名而不是序号对捕获内容进行引用提供了更便捷引用方式不用关注捕获组序号也不用担心表达式部分变更会导致引用捕获组

3.2 非捕获组些表达式中不得不使用( )但又不需要保存( )中子表达式匹配内容这时可以用非捕获组来抵消使用( )带来副作用

表达式	介绍说明
(?:Expression)	进行子表达式Expression匹配并将匹配内容保存到最终整个表达式区配结果中但Expression匹配内容不单独保存到个组内

3.3 反向引用捕获组匹配内容可以在正则表达式外部中进行引用也可以在表达式中进行引用表达式中引用方式就是反向引用

反向引用通常用来查找重复子串或是限定某子串成对出现

表达式	介绍说明
\1\2	

对序号为1和2捕获组反向引用

\k<name>

对命名为name捕获组反向引用

举例:

“(a|b)\1”在匹配“abaa”时匹配成功匹配到结果是“aa” “(a|b)”在尝试匹配时虽然既可以匹配“a”也可以匹配“b”但是在进行反向引用时对应中匹配内容已经是固定了

3.4 环视(Look Around)环视只进行子表达式匹配匹配内容不计入最终匹配结果是零宽度

环视按照方向划分有顺序和逆序两种按照是否匹配有肯定和否定两种组合起来就有 4种环视环视相当于对所在位置加了个附加条件

表达式

介绍说明

(?<=Expression)

逆序肯定环视表示所在位置左侧能够匹配Expression

(?<!Expression)

逆序否定环视表示所在位置左侧不能匹配Expression

(?=Expression)

顺序肯定环视表示所在位置右侧能够匹配Expression

(?!Expression)

顺序否定环视表示所在位置右侧不能匹配Expression

举例:

“(?<=Windows)\d+”在匹配“Windows 2003”时匹配成功匹配结果为“2003”我们知道“\d+”表示匹配个以上数字而“(?<=Windows)”相当于个附加条件表示所在位置左侧必须为“Windows ”它所匹配内容并不计入匹配结果同样正则在匹配“Office 2003”时匹配失败这里任意串数字子串左侧都不是“Windows ”

“(?!1)\d+”在匹配“123”时匹配成功匹配结果为“23” “\d+”匹配个以上数字但是附加条件“(?!1)”要求所在位置右侧不能是“1”所以匹配成功位置是“2”前面位置

3.5 忽略优先和匹配优先或者叫做正则表达式匹配贪婪和非贪婪模式

标准量词修饰子表达式在可匹配可不匹配情况下总会先尝试进行匹配称这种方式为匹配优先或者贪婪模式此前介绍些量词“{m}”、“{m,n}”、“{m,}”、“?”、“*”和“+”都是匹配优先

些NFA正则引擎支持忽略优先量词也就是在标准量词后加个“?”此时在可匹配可不匹配情况下总会先忽略匹配只有在由忽略优先量词修饰子表达式必须进行匹配才能使整个表达式匹配成功时才会进行匹配称这种方式为忽略优先或者非贪婪模式忽略优先量词包括“{m}?”、“{m,n}?”、“{m,}?”、“??”、“*?”和“+?”

举例:

源串: aaa bbb

正则表达式1: .*　　匹配结果: aaa bbb

正则表达式2: .*?　　匹配结果: aaa

2009-9-7 0:38:14
疯狂代码 http://CrazyCoder.cn/