1.正则表达式的用途

**搜索和替换**

**相等**本质上也是搜索操作

2.匹配普通文本

正则表达式可以本身是普通文本。

在正则表达式中，**特殊字符(或字符集合)**用来标示要搜索的东西。

英语句号**.**可以匹配**任意单个**字符，如c.t匹配cat和cot等。

转义字符，需要在原字符前加\，如\.匹配.字符本身

3.匹配一组字符

在正则表达式中，使用元字符**[]**来定义一个**字符集合**，出现在括号中的**所有字符都是该集合的组成部分**，必须**匹配**其中**某个**成员。

[ns]a.\.xls

频繁地用到字符区间(0~9，A~Z)等。可以用**-来定义字符区间**。字符区间是**ASCII**字符表中的字符。区间的首字符一定要**小于**尾字符。-本身是一个特殊的元字符，在**字符集合以外**的地方-不需要被转义。

在同一个字符集合中可以给出多个**字符区间**，如**[A-Za-z0-9]**可以匹配**任何一个**字母或数字。

4.排除

**排除字符集合里指定的字符**，用**^**来排除某个字符集合。

[ns]a[^0-9]\.xls #会舍弃a后面有数字的字符

^效果是作用于给定**字符集合里的所有**字符或字符区间，**不局限于紧跟**的一个字符或字符区间。

5.元字符

元字符是正则表达式中有**特殊含义**的字符，这些字符无法用本身来代替。在元字符的前面加上**\**可对其转义，使其**匹配本身**。

myArray\[0\] #匹配myArray[0]

\本身需要用\\来匹配

6.匹配空白字符的**元字符**

[\b] 表示回退并删除一个字符(Backspace键)

\f 换页符

\n 换行符

\r 回车符

\t 制表符

\v 垂直制表符

**\r\n**匹配**回车+换行**，Windows等操作系统发将这个组合作为文本行的**结束标记**。Linux等使用**\n**作为文本行结束标记。

7.类元字符

**\d** 匹配任意一个数字字符(等价于**[0-9]**)

**\D** 匹配任意一个非数字字符(等价于**[^0-9]**)

**\w** 匹配任意一个字母数字字符(大小写均可)或下划线字符(等价于**[a-zA-Z0-9\_]**)

**\W** 匹配任意一个非字母数字或非下划线字符(等价于**[^a-zA-Z0-9]**)

**\s** 匹配任意一个空白字符(等价于**[\f\n\r\v]**)

**\S** 匹配任意一个非空白字符(等价于**[^\f\n\r\v]**)

十六进制：

正则表达式中**十六进制**用前缀**\x**给出，如**\x0A**对应于ASCII字符10换行符，等价于\n

**八进制**：

前缀为**\0**，数值本身可以是两位或三位数字，**\011**对应于ASCII字符9制表符，等价于\t

**\c**前缀可以制定各种**控制字符**，如**\cZ**匹配**Ctrl-Z**

8.POSIX字符类

POSIX是一种特殊的**标准字符类集**

**[:alnum:]**  匹配任何一个字母或数字(等价于[a-zA-Z0-9])

**[:alpha:]**  匹配任何一个字母(等价于[a-zA-Z])

**[:blank:]** 空格或制表符(等价于[\t ]) (空格用 表示)

**[:cntrl:]** ASCII控制字符(ASCII中0-31和127)

**[:digit:]** 任何一个数字(等价于[0-9])

**[:graph:]** 任何一个可打印字符，不包括空格

**[:lower:]** 任何一个小写字母(等价于[a-z])

**[:print:]** 任何一个可打印字符

**[:punct:]**  不属于[:alnum:]和[:cntrl:]中的任何一个字符

**[:space:]** 任何一个空白字符，等价于[\f\n\r\t\v ]

**[:upper:]** 任何一个大写字母(等价于[:A-Z:])

**[:xdigit:]** 任何一个16进制数字(等价于a-fA-F0-9)

[[:xdigit:]]匹配[0-9A-Fa-f]字符集合，**外层的[]定义一个字符集合，内层的[]是POSIX字符本身的组成部分**

9.匹配一个或多个字符

想要匹配某个字符的一次或多次重复，只需要简单地在其后加上一个**+字符**。**+匹配一个或多个字符**。如a+匹配一个或多个连续出现的a。[0-9]+匹配一个或多个连续的数字。

+是一个元字符，需要用\+来匹配+本身。

**[\w.]+**匹配字母数字字符、下划线、.的一次或多次重复出现。

在字符集合中， .+这样的元字符是否转义的效果都相同。即[\.]等同于[.]

想匹配一个**可有可无**的字符(可以是0个、1个、多个)，需要用**\***元字符。\*用法与+一致，放在某字符或字符集合后，可以匹配该字符**出现0次或多次**的情况。**.\***表示任**意字符出现0次或多次**。

?和+一样，**?**能匹配可选文本，?只能匹配某个字符的**0次或1次**出现，不能超过1次。**https?** 可以匹配http和https

[\r]?和\r?效果一致，只是增加了可读性和避免误解。

1. 由于无法指定具体的匹配次数，正则表达式允许使用**重复范围**。重复范围在**{}**之间指定。

{3}表示匹配前一个字符3次。#[A-Fa-z0-9]{6}表示匹配单个十六进制字符6次。

**{}**语句可以设定**区间范围**，即匹配的**最小次数和最大次数**。{2, 4}表示最少重复2次，最多重复4次。最少重复次数**可以从0开始**。

至少重复多少次

即省略最大值部分，如{3, }表示至少重复3次

11.过度匹配

\*+等都是”贪婪型“元字符，可能会造成过度匹配。其懒惰型版本在匹配时尽可能匹配少的字符，写法是在贪婪型量词后加上一个?

\* 对应 \*? +对应 +? {n, }对应 {n, }?

12.边界

边界是一些用于**指定模式前后位置**的**特殊元字符**。

\bcat\b能够单独匹配cat单词，其中cat不会被包含在其他单词中。

\b表示**单词边界**，匹配一个单词的**开头或结尾**。\b匹配的是**字符之间的一个位置**：一边是单词(能够被\w匹配的字母数字字符和下划线)，另一边是\W能匹配到的内容。\b**不匹配任何实际的字符**，只匹配位置。即**\b的两边中一方是字母**

\B与\b功能相反，不匹配单词边界，可用于查找前后都有多余空格的字符。**\B的两边都不是字母**

13.字符串边界

**字符串边界**元字符有两个： ^表示字符串**开头**，$表示字符串**结尾**

^\s\*<\?xml.\*\?>表示匹配的字符串以空白字符开头

多行模式：

正则表达式中可以使用一些特殊元字符改变另一些元字符，如**(?m)**。它可以启用多行模式，迫使正则表达式引擎将**换行符**视为**字符串分隔符**。这样**^**既可以匹配**字符串开头**，也可以匹配**换行符之后的起始位置**(新行)。**$**可以匹配**字符串结尾**和**换行符之后**的结束位置。

在使用时，(?m)必须出现在整个模式的最前面。

(?m)^\s\*\/\/.\*$ 匹配以//开头的单行注释。如果不加(?m)，则匹配到的字符串会一直延续到代码的末尾。

14.子表达式

**不间断空格**：放在两个单词**之间**的不间断空格能够确保文本**不会在两个单词之间换行，**HTML中用**&nbsp;**表示。如Mr.&nbsp;Smith

&nbsp;{2, }中的{2}只能作用于**紧挨**着它的**前一个字符**，只能匹配&nbsp;;;;。即只作用于;，不匹配&nbsp;nbsp;

划分**子表达式**的目的是将其视为**单一的实体**来使用，子表达式出现在**()**之间，如(&nbsp;){2, }中的{2, }则会作用于2前面的(&nbsp;)。

**|**运算符是**或**操作符，如19|20可以匹配19**或**20。多个|连在一起时，**只要有模式匹配则不会测试选择结构中的其他模式**。

15.反向引用

<[hH][1-6]>.\*?<\/[hH][1-6]> 能匹配<h2>...</h3>，即匹配的第二部分对第一部分**一无所知**，需要使用**反向引用**。反向引用允许正则表达式**引用之前匹配**的结果。

[ ]匹配空格，[ ]+匹配多个空格。

[ ]+(\w+)[ ]+\1 匹配**两个连续且重复**的单词，如of of。原理是(\w+)的**小括号**对该模式进行了**分组**，表达式的最后\1引用了**第一个分组的内容**即(\w+)找出的内容。**若(\w)匹配of，则\1也匹配of**。

同理，\2匹配第二个子表达式，\3匹配第三个子表达式。

16.替换

替换需要**两个子表达式**，一个用于指定**搜索模式**，另一个用来指定**替换模式**。

如正则表达式为**(\w+[\w\.]\*@[\w\.]+\.\w+)**，替换表达式为**<a href=”mailto:$1”>$1</a>** 。其中正则表达式用**()**括起，表示作为**子表达式**。其中替换表达式的**$1**表示正则表达式中的**第一个子表达式**。

**大小写转换**

元字符及说明(用于**替换表达式中**)：

\E 作为\L或\U的**结束符**

\l把**下一个**字符转为小写

\L 把\L到\E之间的字符**全部**转换为小写

\u把**下一个**字符转换为大写

\U把\U到\E之间的字符**全部**转换为大写

如(<[Hh]1>)(.\*?)(<\/[Hh]1>) 替换表达式为$1\U$2\E$3

会将两个<h1>之间的文本转换为大写

17.环视

**环视**可以**标记**匹配的文本位置(而非文本自身)。

**向前查看**

向前查看是**必须匹配但是不用在结果中返回**.向前查看的语法是**以?=开头的子表达式**，需要匹配的文本跟在=后面。**向前查看**是查看出现在已匹配文本**之后**的内容

.+(?=:) 可以匹配到**http**://www.forta.com/中的http。其中.+匹配任意文本，子表达式(?=:)匹配: 。但是:**没有出现在最终的匹配结果中**，**:只是为了向前查看**。也就是说，如果http后不跟随:则不会被匹配到

向后查看

向后查看是查看出现在已匹配文本**之前**的内容，操作符是?<=。

(?<=\$)[0-9.]+ 匹配$**23.45**中的23.45，而不会匹配到$，并且不会匹配到不加$的数字。

向前和向后查看使用时，需要将?<=和内容放入**同一个子组中**。如(?<=[tT][iI][tT][lL][eE]>)则会匹配所有的title。

这些前后查看被称为**肯定式向前查看和肯定式向后查看**。肯定式是指要执行的是匹配操作。

否定式环视

**否定式向前查看**会向前查看**不匹配指定模式**的文本，**否定式向后查看**会向后查看**不匹配指定模式**的文本。用法是!代替原先的=。

如**\b(?<!\$)\d+\b**匹配100、60、50等而**不会匹配带有$**的$30等。单词边界\b的作用是防止匹配$3**0**中的**0**。

18.嵌入式条件

正则表达式中的条件用**?**来定义，如?=和?<=表示肯定式向前/向后查看。**嵌入式条件**语法也使用了?。嵌入式条件处理有**根据反向引用**来进行条件处理、根据**环视**来进行条件处理。

(1反向引用条件

仅在**前一个子表达式匹配**的情况下才允许使用另一个表达式。

将一段文本中的<img>标签全部找出来；如果<img>位于<a>和</a>之间，则也要找出<a>和</a>

**(<[Aa]\s+[^>]+>\s\*)?<[Ii][Mm][Gg]\s+[^>]+>(?(1)\s\*\/[Aa]>)**

**(<[Aa]\s+[^>]+>\s\*)?**匹配一个**<A>或<a>**标签及可能存在的任意属性。

**<[Ii][Mm][Gg]\s+[^>]+>**匹配一个<img>标签及其任意属性。

**(?(1)\s\*\/[Aa]>)**的起始部分是一个条件?(1) 表示仅当第一个反向引用(<A>)标签存在时，才继续匹配**\s\*\/[Aa]>**这些部分。即如果(1)存在，才匹配

(2 环视条件

允许查看**向前或向后查看操作**是否成功来决定要不要执行表达式。

匹配44444-4444的同时不匹配33333-

**\d{5}(?(?=-)-\d{4})**  #这里的**(?(?=-)-\d{4})**是条件。其中?=-表示向前查看一个-，如果连字符存在则不消耗且，则之后的-匹配连字符，d{4}匹配-之后的数字。

19.未完待续

20.

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

34.

35.

36.

37.

38.

39.

40.