正则表达式学习心得

版本：v2.1

Crifan Li

摘要

本文主要介绍了正则表达式的基本语法，以及Python，C#，PHP，Notepad++，Javascript，Perl, Java, EditPlus，UltraEdit，ActionScript，Object-C等语言和工具正则使用心得，以及总结了各个语言间正则表达式的区别。

本文提供多种格式供：

在线阅读	HTML [1]	HTMLs [2]	PDF [3]	CHM [4]	TXT [5]	RTF [6]	WEBHELP [7]
下载（7zip压缩包）	HTML [8]	HTMLs [9]	PDF [10]	CHM [11]	TXT [12]	RTF [13]	WEBHELP [14]

HTML版本的在线地址为：

http://www.crifan.com/files/doc/docbook/regular_expression/release/html/regular_expression.html

有任何意见，建议，提交bug等，都欢迎去讨论组发帖讨论：

http://www.crifan.com/bbs/categories/regular_expression/

修订历史		
修订 2.1	2013-09-05	crl

1. 完成此文逻辑框架
2. 添加了Javascript中的正则表达式
3. 添加了Notepad++t中的正则表达式
4. 添加了Python中的正则表达式的少部分内容
5. 将Python的re的语法和使用心得从language_summary中移过来了
6. 添加了C#的Regex

[1] http://www.crifan.com/files/doc/docbook/regular_expression/release/html/regular_expression.html
[2] http://www.crifan.com/files/doc/docbook/regular_expression/release/htmls/index.html
[3] http://www.crifan.com/files/doc/docbook/regular_expression/release/pdf/regular_expression.pdf
[4] http://www.crifan.com/files/doc/docbook/regular_expression/release/chm/regular_expression.chm
[5] http://www.crifan.com/files/doc/docbook/regular_expression/release/txt/regular_expression.txt
[6] http://www.crifan.com/files/doc/docbook/regular_expression/release/rtf/regular_expression.rtf
[7] http://www.crifan.com/files/doc/docbook/regular_expression/release/webhelp/index.html
[8] http://www.crifan.com/files/doc/docbook/regular_expression/release/html/regular_expression.html.7z
[9] http://www.crifan.com/files/doc/docbook/regular_expression/release/htmls/index.html.7z
[10] http://www.crifan.com/files/doc/docbook/regular_expression/release/pdf/regular_expression.pdf.7z
[11] http://www.crifan.com/files/doc/docbook/regular_expression/release/chm/regular_expression.chm.7z
[12] http://www.crifan.com/files/doc/docbook/regular_expression/release/txt/regular_expression.txt.7z
[13] http://www.crifan.com/files/doc/docbook/regular_expression/release/rtf/regular_expression.rtf.7z
[14] http://www.crifan.com/files/doc/docbook/regular_expression/release/webhelp/regular_expression.webhelp.7z

7. 添加了PHP的PCRE
8. 已将之前Python和C#之间的正则比较整理过来了
9. 已将C#的Regex整理过来了
10. 添加关于Perl, Java, EditPlus，UltraEdit，ActionScript，Object-C的正则
11. 将Python的re整理出去成为独立的教程

正则表达式学习心得:

Crifan Li

版本：v2.1

出版日期 2013-09-05

[15] http://www.crifan.com/files/doc/docbook/soft_dev_basic/release/html/soft_dev_basic.html#cc_by_nc

目录

1. 正则表达式的通用知识 1
    1.1. 正则表达式简介 1
        1.1.1. 什么是正则表达式 1
        1.1.2. 正则表达式的功能和用途 1
            1.1.2.1. 正则表达式的用途举例 1
        1.1.3. 如何使用正则表达式 1
            1.1.3.1. 在某种语言中使用正则表达式 1
            1.1.3.2. 使用专门的正则表达式方面的工具 1
    1.2. 正则表达式的通用语法 2
        1.2.1. 关于贪婪原则和最少原则 4
        1.2.2. 反斜杠 4
        1.2.3. 上尖括号^ 5
        1.2.4. 中括号[]内的+，*，(，) 5
2. Javascript中的正则表达式的学习心得 6
    2.1. javascript中的正则表达式的简介 6
    2.2. javascript正则表达式的语法 6
    2.3. javascript中正则表达式用法举例 6
    2.4. javascript正则表达式的使用心得或注意事项 6
3. Notepad++中的正则表达式的学习心得 8
    3.1. Notepad++中的正则表达式的简介 8
    3.2. Notepad++正则表达式的语法 8
    3.3. Notepad++中正则表达式用法举例 8
4. Python中的正则表达式re模块的学习心得 10
5. C#中的正则表达式的学习心得 11
    5.1. C#中的正则表达式的简介 11
    5.2. C#正则表达式的语法 11
    5.3. C#中的正则表达式的特点 11
    5.4. C#正则表达式的使用心得或注意事项 11
        5.4.1. 正则表达式字符串的写法 11
        5.4.2. 正则表达式中包含双引号，是用两个双引号去表示 11
        5.4.3. C#中的named group和后向引用 11
        5.4.4. 关于字符点，匹配任意字符，但是不匹配换行（\n）的事情 11
            5.4.4.1. 关于默认情况下就是非单行模式，所以默认情况，点是不匹配换行字符\n的 12
            5.4.4.2. 匹配换行\n之外的任意字符，那么也包括了回车\r字符 13
        5.4.5. C#中如何获得带名字name的捕获capture的组group 13
        5.4.6. C#中，对于Regex中带嵌套的括号，那么group是的index如何计算，以及Regex中包含了带name的group，group又是如何计算的。 14
            5.4.6.1. C#中，如果Regex中包括了子括号，即括号嵌套括号的时候，group的index是如何计算的，以及如果Regex中包括了带name的group，group的index是如何计算的 15
6. PHP中的正则表达式的学习心得 18
    6.1. PHP中的正则表达式的简介 18
    6.2. PHP正则表达式的语法 18
    6.3. PHP中正则表达式用法举例 19
    6.4. PHP正则表达式的使用心得或注意事项 20
7. Perl中的正则表达式的学习心得 21
    7.1. Perl中的正则表达式的简介 21
    7.2. Perl正则表达式的语法 21
    7.3. Perl正则表达式用法举例 22
8. Java中的正则表达式的学习心得 23
    8.1. Java中的正则表达式的简介 23
    8.2. Java正则表达式的语法 23
    8.3. Java中正则表达式用法举例 23
    8.4. Java正则表达式的使用心得或注意事项 24

8.4.1. 正则中的斜杠，要写成2个斜杠 24
8.4.2. replaceAll中的斜杠，单个斜杠要写陈四个斜杠 25
8.4.3. java 7之后才支持命名的组(named group) 25
9. Editplus中的正则表达式的学习心得 26
9.1. EditPlus中的正则表达式的简介 26
9.2. EditPlus正则表达式的语法 26
9.3. EditPlus中正则表达式用法举例 26
10. UltraEdit中的正则表达式的学习心得 28
10.1. UltraEdit中的正则表达式的简介 28
10.2. UltraEdit的正则表达式的语法 28
10.3. UltraEdit中正则表达式用法举例 28
10.4. UltraEdit正则表达式的使用心得或注意事项 29
10.4.1. 是"替换"而不是"在文件中替换" 29
10.4.2. Unix版本的正则中的OR操作不支持超过2个的 29
11. ActionScript中的正则表达式的学习心得 30
11.1. ActionScript中的正则表达式的简介 30
11.2. actionscript正则表达式的语法 30
11.3. ActionScript中正则表达式用法举例 30
12. Object-C中的正则表达式的学习心得 31
12.1. Object-C中的正则表达式的简介 31
12.2. Objective-C正则表达式的语法 31
13. 不用语言之间的正则表达式的比较 32
13.1. 一些关于正则表达式中的比较 32
参考书目 36

表格清单

1.1. 正则表达式中通用字符匹配规则 2
1.2. 正则表达式中通用的限定符 3
1.3. 正则表达式中通用的限定符的不同语言中的写法 3
1.4. 正则表达式中已定义的转义字符序列 4
13.1. 不同语言间正则表达式写法的比较 32

范例清单

2.1. Javascript中match的用法举例 6
3.1. Notepad++中用正则表达式实现字符串替换 8
5.1. 默认的点'.'不匹配换行\n 12
6.1. 从html中提取skydrive图片的url 19
7.1. Perl正则去除html的tag 22
8.1. 匹配所有的某种格式的字符串 23
8.2. 带命名的组去替换单个字符串 24
9.1. 查找某个格式的字符串 26
10.1. 查找并更改height和width的值 28

第 1 章 正则表达式的通用知识

1.1. 正则表达式简介

1.1.1. 什么是正则表达式

正则表达式，英文叫做Regular Expression

正则表达式，简单说，就是一组规则，用于实现字符串的查找，匹配，以实现关于字符串的相关操作，比如替换，删除等。

1.1.2. 正则表达式的功能和用途

所谓正则表达式的功能，通俗的说就是，正则表达式能干啥，你能用正则表达式做啥
主要都是针对，字符串，即给定一个字符串，对于一个输入的字符串，然后使用正则表达式，对其进行一定的处理，

目前，个人所见到的，正则表达式的用途，大概可以分为以下几类
判断字符串是否符合某种条件 即所谓的匹配，对应的英文叫做match 从字符串中，提取你想要的内容 即所谓的search 对于提取内容，再细分，有分两类 对于匹配到的，某单个字符串，提取中间的，一个或多个字段 Python中对应的re.search 查找所有的，符合某条件的字符串 Python中对应的re.findall

1.1.2.1. 正则表达式的用途举例

1.1.3. 如何使用正则表达式

简单来说有两种

在某种语言中使用正则表达式

使用专门的正则表达式方面的工具

下面来详细解释一下：

1.1.3.1. 在某种语言中使用正则表达式

正常的用法，绝大多数的用法，都是这类

1.1.3.2. 使用专门的正则表达式方面的工具

另外，也有些，专门的，用于验证和使用正则表达式的工具

比如在线的某些网站，专门提供类似的，验证正则表达式的语法是否正常，功能是否正确

有人专门制作的，正则表达式的学习和使用方面的工具

目前看来，最好用的，算是：RegexBuddy[1]

很多语言目前都已实现了对应的正则表达式的支持。

目前个人已知的有:第 2 章 Javascript中的正则表达式的学习心得, 第 3 章 Notepad++中的正则表达式的学习心得, 第 4 章 Python中的正则表达式re模块的学习心得, 第 5 章 C#中的正则表达式的学习心得, 第 6 章 PHP中的正则表达式的学习心得, 第 7 章 Perl中的正则表达式的学习心得, 第 8 章 Java中的正则表达式的学习心得, 第 9 章 Editplus中的正则表达式的学习心得, 第 10 章 UltraEdit中的正则表达式的学习心得, 第 11 章 ActionScript中的正则表达式的学习心得, 第 12 章 Object-C中的正则表达式的学习心得。

学会了正则表达式后，很多常见的文字，字符串的处理，就简单的多了，或者可以更加高效地实现功能，实现更加复杂的操作了。

入门相对不难，但是能熟练，高效的利用其功能，还是不容易的。

1.2. 正则表达式的通用语法

正则表达式本身，算是一种语言，其有自己一定的规则。

只不过每种其他语言在实现正则表达式的时候，却又对有些规则有些自己的定义。

此处，只是列出一些最基本，各个语言中都通用的一些规则。

表 1.1. 正则表达式中通用字符匹配规则

特殊字符	等价于	含义解释	提示
.		表示任意字符（除了换行符\n以外的任意单个字符）	
^		表示字符串的开始	
$		表示字符串行末或换行符之前	
*	{0,无穷多个}	0或多个前面出现的正则表达式	贪婪原则，即尽量匹配尽可能多个
+	{1,无穷多个}	1或多个前面出现的正则表达式	贪婪原则，即尽量匹配尽可能多个
?			
*?,+?,??		使*，+，?尽可能少的匹配	最少原则
{m}		匹配m个前面出现的正则表达式	
{m,n}		匹配最少m个，最多n个前面出现的正则表达式	贪婪原则，即尽量匹配尽可能多个
{m,n}?		匹配最少m个，最多n个前面前出现的	最少原则
\		后接那些特殊字符，表示该字符；或者后接已经定义好的特殊含义的序列	对于已经定义好的特殊序列，详细解释参考表 1.4 “正则表达式中已定义的转义字符序列”

[1] http://www.regexbuddy.com/index.html

特殊字符	等价于	含义解释	提示
[]		指定所要匹配的字符集，可以列出单个字符；或者是一个字符范围，起始和结束字符之间用短横线-分割；	
|		A|B,其中A和B可以是任意正则表达式，其中也支持更多的抑或A|B|C|...	
(...)		匹配括号内的字符串	实现分组功能，用于后期获得对应不同分组中的字符串

正则中，还有一些，相对比较通用的标记(flag),又称修饰符(modifier)，限定符,或者说不同的模式

表 1.2. 正则表达式中通用的限定符

限定符类型/模式	英文叫法	含义解释
忽略大小写模式	ignore case mode	默认不加此参数，是大小写区分的。加了此参数，则不区分大小写
单行模式	single line mode	默认'.'是不匹配回车换行的。加了此参数，则使得'.'也匹配回车换行，这就使得原本，中间包含了换行的，多行的字符串，变成了单行的效果。
多行模式	multiple line mode	默认情况是：^只匹配，整个（中间包含了回车换行的，多行的）字符串的开始，同理，$只匹配整个字符串的末尾。而加上次参数，则^也匹配每个'\n'==newline=换行符之后的那个位置，表示下一行的开始，同理，$也匹配，换行符之前的那个位置，表示之前一行的结束位置
松散模式	verbose mode/relax regular expression/ ignore Whitespace	默认的正则，都是连续的写在一起的，可以叫做，紧凑型的正则。加上此参数，则运行你在，原先是紧凑的正则中间，加上一些空格或tab，用于对齐格式，以及一些注释内容，目的在于方便别人看懂你写的正则，所以，相对紧凑型的写法，叫做，松散正则。此时，松散正则里面的空白符，即回车，换行，tab，都被忽略，如果想要表达回车换行本身，需要转义。松散正则中的注释，当然在解析的时候，也会被忽略掉的。
区域模式	locale mode	根据当前的local不同，会影响到其他一些正则的含义，比如\w, \W, \b, \B, \s and \S
Unicode模式	unicode mode	根据Unicode字符编码去匹配，会影响到其他一些正则的含义，比如\w, \W, \b, \B, \d, \D, \s and \S

相应的，不同语言中，上述的限定符写法，也不太相同。总结如下

表 1.3. 正则表达式中通用的限定符的不同语言中的写法

不同语言/限定符写法	忽略大小写模式	单行模式	多行模式	松散模式	区域模式	Unicode模式
C#	RegexOptions.IgnoreCase	RegexOptions.Singleline	RegexOptions.Multiline	RegexOptions.IgnorePatternWhitespace	RegexOptions.CultureInvariant（此点待确认）	默认
Pyton	re.IGNORECASE/ re.I	re.DOTALL/ re.S	re.MULTILINE/ re.M	re.VERBOSE/ re.X	re.LOCALE/ re.L	re.UNICODE/ re.U
Perl	i	s	m	x	l	u

表 1.4. 正则表达式中已定义的转义字符序列

特殊序列/已定义的字符含义	等价于	含义解释	提示
\数字		表示前面用括号（ ）括起来的某个组group	
\A		表示字符串的开始	
\b		匹配空字符，但只是，一个单词word的，除了开始或者结束位置的，的空字符	
\B	\b含义取反	0或多个前面出现的正则表达式	
\d	[0-9]	匹配单个数字（0到9中的某个数字）	
\D	[0-9]的取反	非数字的单个字符，即除了0-9之外的任意单个字符	
\s	[\t\n\r\f\v]	匹配单个空白字符，包括空格，水平制表符\t,回车\r,换行\n,换页\f,垂直制表符\v	英文单词一般叫做blankspace，简称space，所以此处的s就是space的意思
\S	\s的含义取反	匹配非空白字符之外的单个字符	
\w	[a-zA-Z0-9_]	匹配单个的任何字符，数字，下划线	
\W	\w的含义取反	匹配单个的，非字母数字下划线之外的字符	
\Z		匹配字符串的末尾	

下面介绍一些通用的规则：

1.2.1. 关于贪婪原则和最少原则

星号*,加号+，问号？，{m,n}，都是属于贪婪原则，即在满足这些条件的情况下，尽可能地匹配多个。

而对应的后面加上一个问号?，变成：*?, +?, ??, {m,n}?就变成了最少原则，即，在满足前面的条件的情况下，尽量少地匹配。

其中??，个人很少用。

而最常用的是 .+?，表示任意字符，最少是1个，然后后面匹配尽量少的所出现的字符。

1.2.2. 反斜杠

在正则表达式中，反斜杠\后面接一些特殊字符，表示该特殊字符，这些特殊是，上面所说过的：

.,^,$,*,+,?

对应的是：

\.,\^,\$,*,\+,\?

以及反斜杠自己：

\\

其中如果是在中括号[]内，反斜杠加上对应的中括号[]和短横线-,即：

\[,\],\-

分别表示对应的字符本身。

1.2.3. 上尖括号^

在中括号之外，表示字符的开始；

在中括号之内，表示取反，比如常见的[^0-9]

比如[^5]表示除了数字5之外的所有字符。

尤其特别的是[^^]表示除了上尖括号字符本身之外的所有字符。

1.2.4. 中括号[]内的+，*，(，)

中括号[]内的加号+，星号*，括号（，）表示对应字符本身，就不是特殊字符了。

第 2 章 Javascript中的正则表达式的学习心得

2.1. javascript中的正则表达式的简介

Javascript中的正则表达式，对应的是对象RegExp。

2.2. javascript正则表达式的语法

详细语法，可以去参考：JavaScript RegExp 对象参考手册[1]

2.3. javascript中正则表达式用法举例

例 2.1. Javascript中match的用法举例

```
//提取skydrive共享地址中的resid（resource id）
shareAddrStr = "https://skydrive.live.com/redir?resid=9A8B8BF501A38A36!1578";
//shareAddrStr = "https://skydrive.live.com/redir.aspx?
cid=9a8b8bf501a38a36&resid=9A8B8BF501A38A36!1578";
var residRe = /resid=(\w+!\d+)/;
var foundResid = shareAddrStr.match(residRe);
if(!foundResid){
    alert("貌似你输入的Skydrive共享地址有误，请确认输入了正确的地址！");
    skydriveShareAddr.focus();
    return false;
}

var resid = foundResid[1];//此处可以获得对应的resid=9A8B8BF501A38A36!1578
```

另外，match返回的结果foundResid，对应的也是foundResid[0]表示所匹配的字符串的整体，此处为：

2.4. javascript正则表达式的使用心得或注意事项

1. 注意两个斜杠最后，不是随便加g或i等属性的
   之前就看到别人说的，正则表达式的具体用法是:

   /xxx/gi

   其中g表示global，i表示区分大小写等等

[1] http://www.w3school.com.cn/js/jsref_obj_regexp.asp

然后我就不小心的在例 2.1 “Javascript中match的用法举例” 中使用了gi属性，结果导致程序运行不正确。

后来才看到RegExp 对象的attributes参数[2]中的解释：

> 是一个可选的字符串，包含属性 "g"、"i" 和 "m"，分别用于指定全局匹配、区分大小写的匹配和多行匹配。
>
> ECMAScript 标准化之前，不支持 m 属性。
>
> 如果pattern是正则表达式，而不是字符串，则必须省略该参数。

而我此处使用match过程中，pattern中是正则表达式，所以，不能加此gi等参数的。

2.

[2] http://www.w3school.com.cn/js/jsref_obj_regexp.asp

第 3 章 Notepad++中的正则表达式的学习心得

3.1. Notepad++中的正则表达式的简介

Notepad++中的正则表达式，指的是搜索，替换时候所用到的正则表达式，去实现复杂的，非规则的，搜索和替换相应的字符串的。

3.2. Notepad++正则表达式的语法

详细语法，可以去参考：How to use regular expressions in Notepad++ (tutorial)[1]

3.3. Notepad++中正则表达式用法举例

例 3.1. Notepad++中用正则表达式实现字符串替换

用如下写法：

查找目标：

```
images/env_var/win/(\w+)\.jpg
```

替换为(P):

```
images/env_var/win/\1\.png
```

可以实现，将：

```
<informalfigure>
   <mediaobject>
      <imageobject role="html"><imagedata fileref="images/env_var/win/
right_click_then_property.jpg" align="left" scalefit="0" width="100%"/></imageobject>
      <imageobject role="fo"><imagedata fileref="images/env_var/win/
right_click_then_property.jpg" align="center" scalefit="1" width="100%"/></imageobject>
   </mediaobject>
</informalfigure>
.....
<informalfigure>
   <mediaobject>
      <imageobject role="html"><imagedata fileref="images/env_var/win/
advance_enviroment.jpg" align="left" scalefit="0" width="100%"/></imageobject>
      <imageobject role="fo"><imagedata fileref="images/env_var/win/
advance_enviroment.jpg" align="center" scalefit="1" width="100%"/></imageobject>
   </mediaobject>
</informalfigure>
```

[1] http://sourceforge.net/apps/mediawiki/notepad-plus/index.php?title=Regular_Expressions

替换为所需要的结果：

```
<informalfigure>
   <mediaobject>
      <imageobject role="html"><imagedata fileref="images/env_var/win/
right_click_then_property.png" align="left" scalefit="0" width="100%"/></imageobject>
      <imageobject role="fo"><imagedata fileref="images/env_var/win/
right_click_then_property.png" align="center" scalefit="1" width="100%"/></imageobject>
   </mediaobject>
</informalfigure>
.....
<informalfigure>
   <mediaobject>
      <imageobject role="html"><imagedata fileref="images/env_var/win/
advance_enviroment.png" align="left" scalefit="0" width="100%"/></imageobject>
      <imageobject role="fo"><imagedata fileref="images/env_var/win/
advance_enviroment.png" align="center" scalefit="1" width="100%"/></imageobject>
   </mediaobject>
</informalfigure>
```

更详细的解释，可以参考：Notepad++的正则表达式替换[2]

[2] http://www.crifan.com/files/doc/docbook/crifan_rec_soft/release/html/crifan_rec_soft.html#npp.regex_replace

第 4 章 Python中的正则表达式re模块的学习心得

已将此部分内容，整理成单独的教程了。详见：

Python专题教程：正则表达式re模块详解 [1]

[1] http://www.crifan.com/files/doc/docbook/python_topic_re/release/html/python_topic_re.html

第 5 章 C#中的正则表达式的学习心得

5.1. C#中的正则表达式的简介

C#中的Regex类处理正则表达式。

5.2. C#正则表达式的语法

5.3. C#中的正则表达式的特点

下面总结一些C#中的正则表达式相对于其他语言中的正则表达式的一些特点，包括优点和缺点：

1. 可以一次性的即搜索多个匹配的整个的字符串，同时又可以一次性地，对于每个字符串，在给定Regex的时候，就分组group，然后得到macthes之后，foreach处理每个match，已经可以得到对应的匹配的结果，可以使用不同group的值了。还是很方便的。
2. 对于匹配多个字符串的时候，好像不能加括号分组的，如果加括号分组了，那么只能匹配单个一个group就结束了。对应的要匹配多个字符串，好像只能使用findall。

5.4. C#正则表达式的使用心得或注意事项

5.4.1. 正则表达式字符串的写法

正则表达式是用字符串前面加上@字符来表示的。

5.4.2. 正则表达式中包含双引号，是用两个双引号去表示

其中如果包含的字符串中包含双引号，那么就两个双引号表示，而不是反斜杠加上双引号（\” ），也不是斜杠加上双引号（/” ）

示例代码：

```
string pat = @"var\s+primedResponse=\{""items""\:\[\{(.*)\}\]\};\s+\$Do\.register
\(""primedResponse""\);";
Regex rx = new Regex(pat,RegexOptions.Compiled);
```

5.4.3. C#中的named group和后向引用

C#中，对于前面所匹配的字符组名的替换和（向后）引用，是不一样的

替换是这样的写法：${name}[1]

而向后引用是这样的写法：\k<name>[2]

5.4.4. 关于字符点，匹配任意字符，但是不匹配换行（\n）的事情

一般对于介绍正则表达式的时候，为了简便，都说成字符点（.），是匹配任何单个字符的。

[1] http://msdn.microsoft.com/en-us/library/ewy2t5e0(v=vs.71).aspx
[2] http://msdn.microsoft.com/en-us/library/thwdfzxy(v=vs.71).aspx

有的解释的严谨的，也会加上一句，在非单行匹配模式下，匹配除了换行\n之外的任意字符。

此句话有更深层的含义：

5.4.4.1. 关于默认情况下就是非单行模式，所以默认情况，点是不匹配换行字符\n的

一般来说，调用的正则表达式的库函数，比如C#中的用new Regex生成对应的Regex变量的时候：

```
Regex imgRx = new Regex(imgP);
MatchCollection foundImg = imgRx.Matches(curSelectCotent);
```

默认就是多行模式，所以如果像我这里，如果去匹配的内容，涉及到中间包括\n字符，那么默认情况下，是无法匹配的，

导致程序调试花了半天时间，才找到上述原因。

例 5.1. 默认的点'.'不匹配换行\n

对于一堆字符：

```
<P><A
href="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles6977DF/F2009082609473229788102 68[2].jpg"><IMG
style="BACKGROUND-IMAGE: none; BORDER-BOTTOM: 0px; BORDER-LEFT: 0px; PADDING-
LEFT: 0px; PADDING-RIGHT: 0px; DISPLAY: inline; BORDER-TOP: 0px; BORDER-RIGHT: 0px;
 PADDING-TOP: 0px"
title=F2009082609473229788102 68 border=0 alt=F2009082609473229788102 68
src="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles6977DF/F2009082609473229788102 68_thumb.jpg"
width=206 height=244></A> </P>
<P><A
href="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles6977DF/王珞丹 图片22[3].jpg"><IMG
style="BACKGROUND-IMAGE: none; BORDER-BOTTOM: 0px; BORDER-LEFT: 0px; PADDING-
LEFT: 0px; PADDING-RIGHT: 0px; DISPLAY: inline; BORDER-TOP: 0px; BORDER-RIGHT: 0px;
 PADDING-TOP: 0px"
title="王珞丹 图片22" border=0 alt="王珞丹 图片22"
src="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles6977DF/王珞丹 图片22_thumb.jpg"
width=176 height=244></A><A
href="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles6977DF/王珞丹 图片333[2].jpg"><IMG
style="BACKGROUND-IMAGE: none; BORDER-BOTTOM: 0px; BORDER-LEFT: 0px; PADDING-
LEFT: 0px; PADDING-RIGHT: 0px; DISPLAY: inline; BORDER-TOP: 0px; BORDER-RIGHT: 0px;
 PADDING-TOP: 0px"
title="王珞丹 图片333" border=0 alt="王珞丹 图片333"
src="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles6977DF/王珞丹 图片333_thumb.jpg"
width=230 height=244></A></P>
```

然后用下面的代码去匹配：

```
string imgP = @"href=""(file:.+?WindowsLiveWriter.+?)"".+?src=""(file:.+?WindowsLiveWriter.
+?)""";
Regex imgRx = new Regex(imgP);
MatchCollection foundImg = imgRx.Matches(curSelectCotent);
```

则其中的"".+?src=""中的.+?是无法匹配成功的，因为上述输入的字符串中，包括了换行符，导致此处无法匹配。

而只有把

```
Regex imgRx = new Regex(imgP);
```

改为:

```
Regex imgRx = new Regex(imgP, RegexOptions.Singleline);
```

才是可以的。

即设置为单行模式，此处点（.）就匹配，包括\n在内的任意单个字符了。

其他语言中，比如Python等，也有对应的single　line，multi　line等方面的设置，可根据自行需要设置。

5.4.4.2. 匹配换行\n之外的任意字符，那么也包括了回车\r字符

所以，万一你的应用中，需要涉及到这个字符，（如果不是单行模式的话），要注意的是点(.)是匹配\r，但不匹配\n的。

5.4.5. C#中如何获得带名字name的捕获capture的组group

对于输入字符串为：

```
<A
href="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles111AD15/王珞丹 图片22[3].jpg"><IMG
style="BACKGROUND-IMAGE: none; BORDER-BOTTOM: 0px; BORDER-LEFT: 0px; PADDING-
LEFT: 0px; PADDING-RIGHT: 0px; DISPLAY: inline; BORDER-TOP: 0px; BORDER-RIGHT: 0px;
 PADDING-TOP: 0px"
title="王珞丹 图片22" border=0 alt="王珞丹 图片22"
src="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles111AD15/王珞丹 图片22_thumb.jpg"
width=176 height=244></A>
```

用下面的代码：

```
string imgP = @"href=""(file:///(?<localAddrPref>.+?WindowsLiveWriter.+?/supfile[^/]+?)/
(?<realName>[^""]+?)\[\d+\](\.(?<suffix>\w{3,4}))?)"".+?src=""(file:///\k<localAddrPref>/
\k<realName>_thumb(\.\k<suffix>)?)""";
```

```
Regex imgRx = new Regex(imgP, RegexOptions.Singleline);
MatchCollection foundImg = imgRx.Matches(curSelectCotent);
```

是可以获得所匹配的group的，但是想要对应带名字的的那些变量，比如localAddrPref，suffix等值，却不知道如何获得。

其实，根据Grouping Constructs[3]的解释，也已经知道如何可以获得，只是觉得那种方法不够好，希望可以找到对应的capture对应的自己的处理方法。

后来找了下，参考Regex: Named Capturing Groups in .NET[4]才明白，原来对于带名字的匹配的group，其实也是group，但是是对于每一个match来说，都有对应的自己的capture的。

结果去试了试，发现如果写成这样：

```
CaptureCollection captures = found.Captures;
localAddrPref   = captures[0].Value;
title           = captures[1].Value; // will lead to code run dead !
```

captures[0]是整个的字符串，而不是对应的第一个name group的值，而captures[1]，则直接导致程序死掉，即不能这样调用。

另外，在Regex: get the name of captured groups in C#[5]也找到了用GetGroupNames的方式，有空也去试试。

看到Regular Expression Classes[6]解释的，可以直接用groupname取得对应的值的，即之前都是用found.Groups[3].ToString()而换作found.Groups["localAddrPref"].Value即可。

其中localAddrPref是前面的某个group的name。

5.4.6. C#中，对于Regex中带嵌套的括号，那么group是的index如何计算，以及Regex中包含了带name的group，group又是如何计算的。

此处的逻辑，是根据官方的解释[7]加上实际调试所得到的结果，来解释说明的。

举例，对于输入字符串：

```
<A
href="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles111AD15/王珞丹 图片22[3].jpg"><IMG
style="BACKGROUND-IMAGE: none; BORDER-BOTTOM: 0px; BORDER-LEFT: 0px; PADDING-
LEFT: 0px; PADDING-RIGHT: 0px; DISPLAY: inline; BORDER-TOP: 0px; BORDER-RIGHT: 0px;
 PADDING-TOP: 0px"
title="王珞丹 图片22" border=0 alt="王珞丹 图片22"
```

[3] http://msdn.microsoft.com/en-us/library/bs2twtah(v=vs.71).aspx
[4] http://stackoverflow.com/questions/906493/regex-named-capturing-groups-in-net
[5] http://stackoverflow.com/questions/1381097/regex-get-the-name-of-captured-groups-in-c-sharp
[6] http://msdn.microsoft.com/en-us/library/30wbz966(v=vs.71).aspx
[7] http://msdn.microsoft.com/en-us/library/bs2twtah(v=vs.71).aspx

```
src="file:///C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/
supfiles111AD15/王珞丹 图片22_thumb.jpg"
width=176 height=244></A>
```

用这样的代码去匹配：

```
string imgP = @"href=""file:///((?<localAddrPref>.+?WindowsLiveWriter.+?/supfile[^/]+?)/
(?<realName>[^""]+?)\[\d+\](\.(?<suffix>\w{3,4}))?)"".+?title=""?(\k<realName>)?""?.+?
alt=""?(\k<realName>)?""?.+?src=""file:///(\k<localAddrPref>/\k<realName>_thumb(\.
\k<suffix>)?)""";
Regex imgRx = new Regex(imgP, RegexOptions.Singleline);
MatchCollection foundImg = imgRx.Matches(curSelectCotent);
```

则匹配的结果是：

```
_groups {System.Text.RegularExpressions.Group[9]} System.Text.RegularExpressions.Group[]
[0] {C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/supfiles111AD15/王珞
丹 图片22[3].jpg} System.Text.RegularExpressions.Group
[1] {.jpg} System.Text.RegularExpressions.Group
[2] {王珞丹 图片22} System.Text.RegularExpressions.Group
[3] {王珞丹 图片22} System.Text.RegularExpressions.Group
[4] {C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/supfiles111AD15/王珞
丹 图片22_thumb.jpg} System.Text.RegularExpressions.Group
[5] {.jpg} System.Text.RegularExpressions.Group
[6] {C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/supfiles111AD15}
 System.Text.RegularExpressions.Group
[7] {王珞丹 图片22} System.Text.RegularExpressions.Group
[8] {jpg} System.Text.RegularExpressions.Group
```

注意，上述内容是debug所查看到的。

group[0]不是整个字符串，而是表示此处匹配的第一个结果，即index从0开始，而不是默认的1.

不过下面还是以代码中的写法来解释。

5.4.6.1. C#中，如果Regex中包括了子括号，即括号嵌套括号的时候，group的index是如何计算的，以及如果Regex中包括了带name的group，group的index是如何计算的

搞了半天，最后终于看懂了。

现在来说说，如何数group的index。

其实很简单:

1. 首先，所有的group，分两类，一类是默认的，无名字（unnamed的group），另外一类是有名字的group（name的group）。
2. 然后对于group的index，从左至右，以左括号出现的位置先后为标准，一个个的数一下对应的unnamed的group，等全部数完了，再去安装同样标准去数有名字的group。

3. 或者可以完全不去数有name的group，因为有name的group的值，完全可以直接通过Groups["yorGroupName"].Value的形式得到。

举例说明：

对于例子中的匹配样式：

```
href=""file:///((?<localAddrPref>.+?WindowsLiveWriter.+?/supfile[^/]+?)/(?<realName>[^""]+?)\[\d+\](\.(?<suffix>\w{3,4}))?)"".+?title=""?(\k<realName>)?""?.+?alt=""?(\k<realName>)?""?.+?src=""file:///(\k<localAddrPref>/\k<realName>_thumb(\.\k<suffix>)?)""
```

从左往右，第一个是file:///后面的左括号，即对应的括号是?)"".+?title中的这个右括号，

由于其是unnamed的group，所以代码中的index为第一个，为1,

对应着代码和值是：

Groups[1].ToString() == 上述调试的结果 C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/supfiles111AD15/王珞丹 图片22[3].jpg

然后再往右数，第二个括号是(?<localAddrPref>中的左括号，对应右括号是+?)/(?<realName>中的右括号，

由于其实有名字的group，所以对应的index，不是Groups[2]，而是等最后数完了全部的unnamed的group，才能知道此处的index。

但是由于是有名字的，所以也是可以直接通过group的name来获得对应的值的。

对应的代码和值是：

Groups["localAddrPref"] == C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/supfiles111AD15

然后按照此法：

从左往右，依次按照左括号出现的先后顺序去数，得到对应的各个group是：

1. 第一个： ((?<localAddrPref>.+?WindowsLiveWriter.+?/supfile[^/]+?)/(?<lrealName>[^""]+?)\[\d+\](\.(?<lsuffix>\w{3,4}))?)
   unnamed index为1，值为Groups[1].ToString()
2. 第二个: (?<localAddrPref>.+?WindowsLiveWriter.+?/supfile[^/]+?)
   有名字，index待定，值为Groups["localAddrPref"].Value
3. 第三个： (?<realName>[^""]+?)
   有名字，index待定，值为Groups["realName"].Value
4. 第四个: (\.(?<suffix>\w{3,4}))
   unnamed，index为2，值为Groups[2].ToString()
5. 第五个：(?<suffix>\w{3,4})
   有名字，index待定，值为Groups["suffix"].Value
6. 第六个： (\k<realName>)
   unnamed，index为3，值为Groups[3].ToString()
7. 第七个： (\k<realName>)

unnamed，index为4，值为Groups[4].ToString()

8. 第八个：(\k<localAddrPref>/\k<realName>_thumb(\.\k<suffix>)?)
   unnamed，index为5，值为Groups[5].ToString()

9. 第九个：(\.\k<suffix>)
   unnamed，index为6，值为Groups[6].ToString()

对照着上述debug出来的结果，正好分别就是：

1. Groups[1].ToString() == debug中的[0] == C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/supfiles111AD15/王珞丹 图片22[3].jpg

2. Groups[2].ToString() == debug中的[1] == .jpg

3. Groups[3].ToString() == debug中的[2] == 王珞丹 图片22

4. Groups[4].ToString() == debug中的[3] == 王珞丹 图片22

5. Groups[5].ToString() == debug中的[4] == C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/supfiles111AD15/王珞丹 图片22_thumb.jpg

6. Groups[6].ToString() == debug中的[5] == .jpg

而unnamed的全部都数完了，然后才是从左到右的，有名字的index依次增加，以及对应的值是：

1. Groups["localAddrPref"].Value == Groups[7].ToString() == debug中的[6] == C:/Users/CLi/AppData/Local/Temp/WindowsLiveWriter1627300719/supfiles111AD15

2. Groups["realName"].Value == Groups[8].ToString() == debug中的[7] == 王珞丹 图片22

3. Groups["suffix"].Value == Groups[9].ToString() == debug中的[8] == jpg

所以，对于Regex中的group和named的group的index如何计算，就是上面那一句话：

从左到右，以左括号出现位置先后为顺序，index一次增加。

先全部数完unnamed的group，再去数有name的group，其中有name的group，完全可以不去计算对应的index是多少的，因为可以直接通过名字来引用对应的值，用法为：

Groups["yorGroupName"].Value

第 6 章 PHP中的正则表达式的学习心得

6.1. PHP中的正则表达式的简介

PHP中的正则表达式，总称为PCRE系列函数[1]，对应的是一堆以preg_开头的函数：

- preg_match
  执行一个正则表达式匹配
- preg_match_all
  执行一个全局正则表达式匹配
- preg_replace
  执行一个正则表达式的搜索和替换
- preg_split
  通过一个正则表达式分隔字符串
- preg_filter
  执行一个正则表达式搜索和替换
- preg_grep
  返回匹配模式的数组条目
- preg_last_error
  返回最后一个PCRE正则执行产生的错误代码
- preg_quote
  转义正则表达式字符
- preg_replace_callback
  执行一个正则表达式搜索并且使用一个回调进行替换

6.2. PHP正则表达式的语法

关于PHP中的语法，所有内容都可以在这里找到Regular Expressions (Perl-Compatible)[2]

下面只是摘录相关的链接，以方便有需要的，直接点击对应链接看自己所关心的内容：

- 简介 [3]
- 分隔符 [4]
- 元字符 [5]
- 转义序列(反斜线) [6]
- Unicode字符属性 [7]
- 锚 [8]

[1] http://www.php.net/manual/zh/ref.pcre.php
[2] http://cn.php.net/pcre
[3] http://cn.php.net/manual/zh/regexp.introduction.php
[4] http://cn.php.net/manual/zh/regexp.reference.delimiters.php
[5] http://cn.php.net/manual/zh/regexp.reference.meta.php
[6] http://cn.php.net/manual/zh/regexp.reference.escape.php
[7] http://cn.php.net/manual/zh/regexp.reference.unicode.php
[8] http://cn.php.net/manual/zh/regexp.reference.anchors.php

- 句点 [9]
- 字符类(方括号) [10]
- 可选路径(|) [11]
- 内部选项设置 [12]
- 子组(子模式) [13]
- 重复/量词 [14]
- 后向引用 [15]
- 断言 [16]
- 一次性子组 [17]
- 条件子组 [18]
- 注释 [19]
- 递归模式 [20]
- 性能 [21]

6.3. PHP中正则表达式用法举例

例 6.1. 从html中提取skydrive图片的url

想要从一段html中提取出这样的地址：

http://storage.live.com/items/9A8B8BF501A38A36!1211?filename=%e7%82%b9%e5%87%bb%e7%ac%ac%e5%87%a0%e9%a1%b5.jpg

然后对应的php代码为：

```
# current support valid pic suffiex: 'bmp', 'gif', 'jpeg', 'jpg', 'jpe', 'png', 'tiff', 'tif'
$picSufChars = "befgijmnptBEFGIJMNPT";
//$pattern = '#<a href="(?P<picUrl>http://storage.live.com/items/.+?filename=.+?\.jpg)"><img.+?src="(?P=picUrl)".+?/></a>#';
$pattern = '#<a href="(?P<picUrl>http://storage.live.com/items/.+?filename=.+?\.['.$picSufChars.']{3,4})"><img.+?src="(?P=picUrl)".+?/></a>#';

if(preg_match($pattern, $content, $matches)){
    $picUrl = $matches["picUrl"];
```

[9] http://cn.php.net/manual/zh/regexp.reference.dot.php
[10] http://cn.php.net/manual/zh/regexp.reference.character-classes.php
[11] http://cn.php.net/manual/zh/regexp.reference.alternation.php
[12] http://cn.php.net/manual/zh/regexp.reference.internal-options.php
[13] http://cn.php.net/manual/zh/regexp.reference.subpatterns.php
[14] http://cn.php.net/manual/zh/regexp.reference.repetition.php
[15] http://cn.php.net/manual/zh/regexp.reference.back-references.php
[16] http://cn.php.net/manual/zh/regexp.reference.assertions.php
[17] http://cn.php.net/manual/zh/regexp.reference.onlyonce.php
[18] http://cn.php.net/manual/zh/regexp.reference.conditional.php
[19] http://cn.php.net/manual/zh/regexp.reference.comments.php
[20] http://cn.php.net/manual/zh/regexp.reference.recursive.php
[21] http://cn.php.net/manual/zh/regexp.reference.performance.php

```
    //print "found picUrl=".$picUrl;
    if(preg_match('#http://storage\.live\.com/items/.+?filename=(?P<filename>.+)#', $picUrl,
 $foundFilename)){
        $filename = $foundFilename["filename"];
        $filename = rawurldecode($filename);
    }
    else{
        $filename = "";
    }

} else {
    $picUrl = "";
}
```

详细代码可参考【已解决】给Retina 0.2中添加支持日志缩略图中显示文章中所包含的第一张图片[22]

6.4. PHP正则表达式的使用心得或注意事项

[22] http://www.crifan.com/wordpress_theme_retina_add_support_show_thumbnail_for_first_skydrive_picture_in_content_html/

第 7 章 Perl中的正则表达式的学习心得

7.1. Perl中的正则表达式的简介

Perl中的正则表达式，最主要的语法就两种：m//和s///

- m//
  m=match,表示匹配
- s///
  s=substitution，表示替换

不论是匹配还是替换，后面都运行加上一些限制参数，叫做修饰符（Modifiers）

- m
  m=multiple line=多行模式
- s
  s=single line=单行模式
- i
  i=ignore case=忽略大小写
- x
  x=extended=扩展模式=允许白空格和注释=松散正则

7.2. Perl正则表达式的语法

关于Perl中的正则表达式的语法，目前找到的，相对最完整的介绍，在这里：perlre[1]

下面把其目录摘录如下，供快速查阅相关内容：

- Modifiers [2]
- Regular Expressions [3]
- Quoting metacharacters [4]
- Extended Patterns [5]
- Special Backtracking Control Verbs [6]
- Backtracking [7]
- Version 8 Regular Expressions [8]
- Warning on \1 Instead of $1 [9]
- Repeated Patterns Matching a Zero-length Substring [10]

[1] http://perldoc.perl.org/perlre.html
[2] http://perldoc.perl.org/perlre.html#Modifiers
[3] http://perldoc.perl.org/perlre.html#Regular-Expressions
[4] http://perldoc.perl.org/perlre.html#Quoting-metacharacters
[5] http://perldoc.perl.org/perlre.html#Extended-Patterns
[6] http://perldoc.perl.org/perlre.html#Special-Backtracking-Control-Verbs
[7] http://perldoc.perl.org/perlre.html#Backtracking
[8] http://perldoc.perl.org/perlre.html#Version-8-Regular-Expressions
[9] http://perldoc.perl.org/perlre.html#Warning-on-%5c1-Instead-of-%241
[10] http://perldoc.perl.org/perlre.html#Repeated-Patterns-Matching-a-Zero-length-Substring

- Combining RE Pieces [11]
- Creating Custom RE Engines [12]
- PCRE/Python Support [13]

7.3. Perl正则表达式用法举例

例 7.1. Perl正则去除html的tag

想要将html

```
<h1>h1 content</h1>
<div>
   div test
</div>

<invalidTag> invalid tag test </invalid>
```

中的标签tag去掉，变成：

```
h1 content

   div test


<invalidTag> invalid tag test </invalid>
```

用的perl的正则的代码是：

```
$filteredHtml =~  s/<(\w+?)>(.+?)<\/\1>/$2/sg;
```

详细代码可参考【已解决】Perl中的正则表达式的替换和后向引用[14]

[11] http://perldoc.perl.org/perlre.html#Combining-RE-Pieces
[12] http://perldoc.perl.org/perlre.html#Creating-Custom-RE-Engines
[13] http://perldoc.perl.org/perlre.html#PCRE%2fPython-Support
[14] http://www.crifan.com/perl_regex_replace_backreference/

第 8 章 Java中的正则表达式的学习心得

8.1. Java中的正则表达式的简介

Java中处理正则表达式的类是：java.util.regex

其下主要有两个大的子类：

- java.util.regex.Pattern
  专门用于处理正则表达式这个字符串本身
- java.util.regex.Matcher
  处理，找到匹配的项之后，后续的处理，比如单个的替换，替换所有的等等

8.2. Java正则表达式的语法

Java中，正则表达式的语法，可以在官网文档中找到：

- Java 6的正则表达式的语法
  (Java 6) java.util.regex.Pattern[1]

  (Java 6) java.util.regex.Matcher[2]
- Java 7的正则表达式的语法
  (Java 7) java.util.regex.Pattern[3]

  (Java 7) java.util.regex.Matcher[4]

8.3. Java中正则表达式用法举例

例 8.1. 匹配所有的某种格式的字符串

想要匹配：

11A11、22A22、33A33、44B44、55B55

中的每个值，并打印出来，然后对应的Java代码为：

```
String digitNumStr = "11A11、22A22、33A33、44B44、55B55";
//String digitNumStr = "11A11";
Pattern digitNumP = Pattern.compile("(?<twoDigit>\\d{2})[A-Z]\\k<twoDigit>");
Matcher foundDigitNum = digitNumP.matcher(digitNumStr);

// Find all matches
while (foundDigitNum.find()) {
  // Get the matching string
  String digitNumList = foundDigitNum.group();
  System.out.println(digitNumList);
}
```

[1] http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html
[2] http://docs.oracle.com/javase/6/docs/api/java/util/regex/Matcher.html
[3] http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
[4] http://docs.oracle.com/javase/7/docs/api/java/util/regex/Matcher.html

详细代码可参考【已解决】Java的正则表达式Regex中，如何查找所有的匹配的项[5]

例 8.2. 带命名的组去替换单个字符串

想要将字符串

```
aa/haha.exe
```

中的前面部分去掉，只保留后面部分，即：

```
haha.exe
```

则用的java代码为：

```
String filenameStr = "aa/haha.exe";

//named group only support after Java 7
//here is my java version:
//Java: 1.7.0_09; Java HotSpot(TM) 64-Bit Server VM 23.5-b02
Pattern filenameP = Pattern.compile("^.+/(?<filenamePart>.+)$");
Matcher filenameMatcher = filenameP.matcher(filenameStr);
boolean foundFilename = filenameMatcher.matches();

System.out.println(filenameMatcher);
System.out.println(foundFilename);

String onlyFilename = filenameMatcher.replaceFirst("${filenamePart}");
System.out.println(onlyFilename);
```

输出为：

```
java.util.regex.Matcher[pattern=^.+/(?<filenamePart>.+)$ region=0,11 lastmatch=aa/
hahatrue]
true
haha.exe
```

详细代码可参考：

【已解决】Java中的正则表达式（java.util.regex）的替换 [6]

【已解决】Java的正则表达式java.util.regex中的命名的组（named group） [7]

8.4. Java正则表达式的使用心得或注意事项

8.4.1. 正则中的斜杠，要写成2个斜杠

普通程序中，即使是正则中，斜杠也就是斜杠。

[5] http://www.crifan.com/java_regex_how_to_findall/
[6] http://www.crifan.com/java_util_regex_replacement/
[7] http://www.crifan.com/java_util_regex_named_group/

但是java中，由于string的设计，导致斜杠，是特殊的转义字符，所以，在正则中，如果想要写普通的，正则的转义，比如'\d'表示数字，则要写成'\\d'才可以。

所就变成了：其他程序中，正常的写单个的斜杠的，java中，都要变成双斜杠。

详见：【已解决】Java的正则表达式java.util.regex中匹配星号’ *’ asterisk字符本身[8]

8.4.2. replaceAll中的斜杠，单个斜杠要写陈四个斜杠

如上所说，java中的，正则中的斜杠字符本身，要写成两个斜杠

而如果遇到replaceAll等函数，则一个斜杠，要写成四个斜杠，才能被识别。

比如你想把字符串中，所有的，的单个斜杠，都变成双斜杠，则要写成

```
string.replaceAll("\\\\", "\\\\\\\\");
```

详见：【已解决】android中的java的正则的替换：去除掉宏定义中行末的反斜杠[9]

8.4.3. java 7之后才支持命名的组(named group)

即，Java 6之前不支持named group

详见：【已解决】Java的正则表达式java.util.regex中的命名的组（named group）[10]

[8] http://www.crifan.com/java_regex_match_asterisk_char_itself/
[9] http://www.crifan.com/android_java_string_regex_replace_remove_slash_in_multiple_define_line_end/
[10] http://www.crifan.com/java_util_regex_named_group/

第 9 章 Editplus中的正则表达式的学习心得

9.1. EditPlus中的正则表达式的简介

EditPlus中的正则表达式，算是有两种：

- 没有选中"Use TR1 regular expression"
  正则的功能，是支持，但是极度的弱，不好用。
- 选中"Use TR1 regular expression"
  即，使用C++的TR1的正则，简单说就是，从原先的，功能很弱的正则，升级到功能相对正常的正则，支持常见的正则的语法了

9.2. EditPlus正则表达式的语法

如果想要使用，正常的正则，需要去开启TR1选项：

Tools->Preferences->General -> Use TR1 regular expression

然后就可以正常的使用正则了。

不过，官网中也没给出相关的TR1的语法。所以，只能靠自己，对于常见的正则的语法的了解去写正则了。

但是关于TR1的解释，倒是可以参考微软的官网解释：TR1 Regular Expressions[1]

9.3. EditPlus中正则表达式用法举例

例 9.1. 查找某个格式的字符串

想要从：

```
up/114567/img
up/120305/img
up/1205/img
```

中，查找12开头、数字为4位的字符，即：

```
up/1205/img
```

在选中了上面那个"Use TR1 regular expression"的前提下，用的正则是：

[1] http://msdn.microsoft.com/en-us/library/bb982727.aspx

```
up/12\d{2}/img
```

详细代码可参考【吐槽】Editplus中的正则表达式，功能虽然不垃圾，但相关的文档却很垃圾[2]

[2] http://www.crifan.com/editplus_regular_expression_function_is_too_weak_and_has_bug/

第 10 章 UltraEdit中的正则表达式的学习心得

10.1. UltraEdit中的正则表达式的简介

UltraEdit中的正则表达式，支持三种：

- Perl
  用Perl的正则表达式的语法
- Unix
  用常见的标准的正则的语法
- UltraEdit
  用UltraEdit自己的一套的正则的语法

10.2. UltraEdit的正则表达式的语法

UltraEdit中的正则，支持三种语法，全部都可以在官网页面中找到具体的语法的解释：

- Unix和UltraEdit版本的正则的语法
  Regular expressions[1]
- Perl版本的正则的语法
  Perl regular expressions: Getting started[2]

  Perl regular expressions: Non-greedy regex[3]

  Perl regular expressions: Digging deeper[4]

  Perl regular expressions: Backreferences[5]

10.3. UltraEdit中正则表达式用法举例

例 10.1. 查找并更改height和width的值

对于如下内容：

```
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 0
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 1
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 2
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 3
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 4
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 5
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 6
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 7
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 8
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 9
```

[1] http://www.ultraedit.com/support/tutorials_power_tips/ultraedit/regular_expressions.html
[2] http://www.ultraedit.com/support/tutorials_power_tips/ultraedit/perl-regular-expressions.html
[3] http://www.ultraedit.com/support/tutorials_power_tips/ultraedit/non-greedy-perl-regex.html
[4] http://www.ultraedit.com/support/tutorials_power_tips/ultraedit/perl-regular-expressions-digging-deeper.html
[5] http://www.ultraedit.com/support/tutorials_power_tips/ultraedit/perl-regular-expressions-backreferences.html

```
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 10
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 11
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 12
```

想要实现：只想要处理instance后面是0 4 8 12的行，将720替换成576，1280替换成1024

则，在UltraEdit中，需要：

正则的语法，注意要选择为：Perl，而不能选择为Unix，其原因，详见：【整理】UltraEdit中的Unix版本的正则竟然不支持超过2个或者关系[6]

然后查找和替换的正则分别为：

```
-height\s+\d+\s+-width\s+\d+\s+-vidformat\s+1\s+-instance\s+(0|2|4|8)
-height 576 -width 1024 -vidformat 1 -instance \1
```

就可以替换为：

```
./xcode.configVpp.pl -frmrate 5 -height 576 -width 1024 -vidformat 1 -instance 0
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 1
./xcode.configVpp.pl -frmrate 5 -height 576 -width 1024 -vidformat 1 -instance 2
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 3
./xcode.configVpp.pl -frmrate 5 -height 576 -width 1024 -vidformat 1 -instance 4
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 5
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 6
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 7
./xcode.configVpp.pl -frmrate 5 -height 576 -width 1024 -vidformat 1 -instance 8
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 9
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 10
./xcode.configVSC.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 11
./xcode.configVpp.pl -frmrate 5 -height 720 -width 1280 -vidformat 1 -instance 12
```

10.4. UltraEdit正则表达式的使用心得或注意事项

10.4.1. 是"替换"而不是"在文件中替换"

普通常见的要去做替换操作，处理当前文件中的内容，则是从：搜索->替换，打开的替换的对话框，而不是：搜索->在文件中替换。

否则每次去替换，还会提示你"在文件中替换可能会修改驱动器上的许多文件"

详见：【整理】UltraEdit中的Unix版本的正则竟然不支持超过2个或者关系[7]

10.4.2. Unix版本的正则中的OR操作不支持超过2个的

用Unix的正则时，OR操作，不支持超过2个的。

详见：【整理】UltraEdit中的Unix版本的正则竟然不支持超过2个或者关系[8]

[6] http://www.crifan.com/ultraedit_unix_version_regex_not_support_more_than_two_item_do_or
[7] http://www.crifan.com/ultraedit_unix_version_regex_not_support_more_than_two_item_do_or
[8] http://www.crifan.com/ultraedit_unix_version_regex_not_support_more_than_two_item_do_or

第 11 章 ActionScript中的正则表达式的学习心得

11.1. ActionScript中的正则表达式的简介

ActionScript中的正则表达式，对应的是对象RegExp。

其和javascript很类似。

根据[9]的解释，ActionScript是Adobe为Flash开发的脚步语言，是基于ECMAScript的。

而javascript也是基于ECMAScript的，所以两者，至少在正则方面，那基本上都是同一个regexp的库了，都是差不多的。

11.2. actionscript正则表达式的语法

ActionScript的正则的语法，简述为；

ActionScript的正则，是RegExp类，其主要包含两个方法：exec() 和 test()。

可以参考Adobe官网的，详细解释：对字符串使用正则表达式的方法[1]

11.3. ActionScript中正则表达式用法举例

暂无。

可参考javascript中的例子：第 2.3 节 “javascript中正则表达式用法举例”

[1] http://help.adobe.com/zh_CN/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7ea8.html

第 12 章 Object-C中的正则表达式的学习心得

12.1. Object-C中的正则表达式的简介

Objective-C　Cocoa中的正则表达式，起初，是没有内置的库支持的。所以，很多人常用第三方面的库：RegexKitLite

而后来在OS X v10.7/iPhone 4.0后，有了内置支持正则的库：NSRegularExpression

12.2. Objective-C正则表达式的语法

关于RegexKitLite可以（要翻墙）参考其官网：RegexKitLite[1]

关于内置的库NSRegularExpression，参考mac官网：NSRegularExpression Class Reference[2]

[1] http://regexkit.sourceforge.net/RegexKitLite/

[2] http://developer.apple.com/library/mac/#documentation/Foundation/Reference/NSRegularExpression_Class/Reference/Reference.html

第 13 章 不用语言之间的正则表达式的比较

13.1. 一些关于正则表达式中的比较

旧帖

此部分的内容的旧帖为：【总结】关于（C#和Python中的）正则表达式[1]

下面对于各种语言之间的正则表达式的语法，进行简单的比较：

表 13.1. 不同语言间正则表达式写法的比较

	Python	C#	PHP	Javascript	Notepad++
Named Group	(?P<groupName>xxx)	(?<groupName>xxx)❶	(?<groupName>xxx)❷		
正则表达式字符串的表示方法	r"(.+?)"	@"(.+?)"			
向前匹配（prev match）	(?<=xxx)❸	(?<=xxx)❹			
向后匹配（post match）	(?=xxx)❸	(?=xxx)❹			
向前一定不要匹配（prev non-match）	(?<!xxx)❸	(?<!xxx)❹			
向后一定不要匹配（post non-match）	(?<=xxx)❸	(?<=xxx)❹			

❶ 示例代码为：

```
import re;

#--------------------------------------------------------------------------------
def testBackReference():
    # back reference (?P=name) test
    backrefValidStr =
 '"group":0,"iconType":"NonEmptyDocumentFolder","id":"9A8B8BF501A38A36!
601","itemType":32,"name":"released","ownerCid":"9A8B8BF501A38A36"';
    backrefInvalidStr =
 '"group":0,"iconType":"NonEmptyDocumentFolder","id":"9A8B8BF501A38A36!
601","itemType":32,"name":"released","ownerCid":"987654321ABCDEFG"';
    backrefP = r'"group":\d+,"iconType":"\w+","id":"(?P<userId>\w+)!\d+","itemType":\d
+,"name":".+?","ownerCid":"(?P=userId)"'
    userId = "";
```

[1] http://www.crifan.com/summary_regular_expression_csharp_python/

```
   foundBackref = re.search(backrefP, backrefValidStr);
   print "foundBackref=",foundBackref; # foundBackref= <_sre.SRE_Match object at
0x02B96660>
   if(foundBackref):
      userId = foundBackref.group("userId");
      print "userId=",userId; # userId= 9A8B8BF501A38A36
      print "can found userId here";

   foundBackref = re.search(backrefP, backrefInvalidStr);
   print "foundBackref=",foundBackref; # foundBackref= None
   if(not foundBackref):
      print "can NOT found userId here";

   return ;
```

❷ 示例代码为：

```
using System.Text.RegularExpressions;

void testBackReference()
{
   // back reference \k<name> test
   string backrefValidStr = "\"group\":0,\"iconType\":\"NonEmptyDocumentFolder\",
\"id\":\"9A8B8BF501A38A36!601\",\"itemType\":32,\"name\":\"released\",\"ownerCid\":
\"9A8B8BF501A38A36\"";
   string backrefInvalidStr = "\"group\":0,\"iconType\":\"NonEmptyDocumentFolder\",
\"id\":\"9A8B8BF501A38A36!601\",\"itemType\":32,\"name\":\"released\",\"ownerCid\":
\"987654321ABCDEFG\"";
   string backrefP = @""group"":\d+,""iconType"":""\w+"",""id"":""(?<userId>\w+)!\d
+"",""itemType"":\d+,""name"":"".+?"",""ownerCid"":""\k<userId>""";
   string userId = "";
   Regex backrefValidRx = new Regex(backrefP);
   Match foundBackref;

   foundBackref = backrefValidRx.Match(backrefValidStr);
   if (foundBackref.Success)
   {
      // can found the userId
      userId = foundBackref.Groups["userId"].Value;
      MessageBox.Show("can found the userId !");
   }

   foundBackref = backrefValidRx.Match(backrefInvalidStr);
   if (foundBackref.Success)
   {
      // can NOT found the userId
   }
   else
   {
      MessageBox.Show("can NOT found the userId !");
   }
}
```

❸ 示例代码为：

```
import re;

#--------------------------------------------------------------------------
def testPrevPostMatch():
   # post match:       (?=xxx)
   # post non-match:   (?!xxx)
   # prev match:       (?<=xxx)
   # prev non-match:   (?<!xxx)

   #note that input string is:
   #src=\"http://b101.photo.store.qq.com/psb?/
V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA\"
   qqPicUrlStr             = 'src=\\"http://b101.photo.store.qq.com/psb?/
V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA\\"';
   qqPicUrlInvalidPrevStr  = '1234567http://b101.photo.store.qq.com/psb?/
V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA\\"';
   qqPicUrlInvalidPostStr  = 'src=\\"http://b101.photo.store.qq.com/psb?/
V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA123';
   canFindPrevPostP = r'(?<=src=\\")(?P<qqPicUrl>http://.+?\.qq\.com.+?)(?=\\")';
   qqPicUrl = "";

   foundPrevPost = re.search(canFindPrevPostP, qqPicUrlStr);
   print "foundPrevPost=",foundPrevPost; #
   if(foundPrevPost):
      qqPicUrl = foundPrevPost.group("qqPicUrl");
      print "qqPicUrl=",qqPicUrl; # qqPicUrl= http://b101.photo.store.qq.com/
psb?/V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA
      print "can found qqPicUrl here";

   foundInvalidPrev = re.search(canFindPrevPostP, qqPicUrlInvalidPrevStr);
   print "foundInvalidPrev=",foundInvalidPrev; # foundInvalidPrev= None
   if(not foundInvalidPrev):
      print "can NOT found qqPicUrl here";

   foundInvalidPost = re.search(canFindPrevPostP, qqPicUrlInvalidPostStr);
   print "foundInvalidPost=",foundInvalidPost; # foundInvalidPost= None
   if(not foundInvalidPost):
      print "can NOT found qqPicUrl here";

   return ;
```

❹ 示例代码为：

```
using System.Text.RegularExpressions;

void testPrevPostMatch()
```

```
{
   //http://msdn.microsoft.com/en-us/library/bs2twtah(v=vs.71).aspx
   // post match:       (?=xxx)
   // post non-match:   (?!xxx)
   // prev match:       (?<=xxx)
   // prev non-match:   (?<!xxx)

   //src=\"http://b101.photo.store.qq.com/psb?/
V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA\"
   string qqPicUrlStr              = "src=\\\"http://b101.photo.store.qq.com/psb?/
V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA\\\"";
   string qqPicUrlInvalidPrevStr   = "12345678http://b101.photo.store.qq.com/
psb?/V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA\\\"";
   string qqPicUrlInvalidPostStr   = "src=\\\"http://b101.photo.store.qq.com/
psb?/V10ppwxs00XiXU/5dbOIlYaLYVPWOz*1nHYeSFq09Z5rys72RIJszCsWV8!/b/
YYUOOzy3HQAAYqsTPjz7HQAA1234";
   string canFindPrevPostP = @"(?<=src=\\"")(?<qqPicUrl>http://.+?\.qq\.com.+?)(?=\
\"")";
   string qqPicUrl = "";

   Regex prevPostRx = new Regex(canFindPrevPostP);

   Match foundPrevPost = prevPostRx.Match(qqPicUrlStr);
   if (foundPrevPost.Success)
   {
      qqPicUrl = foundPrevPost.Groups["qqPicUrl"].Value;
      MessageBox.Show("can found the qqPicUrl !");
   }

   Match foundInvalidPrev = prevPostRx.Match(qqPicUrlInvalidPrevStr);
   if (!foundInvalidPrev.Success)
   {
      MessageBox.Show("can NOT found the qqPicUrl !");
   }

   Match foundInvalidPost = prevPostRx.Match(qqPicUrlInvalidPostStr);
   if (!foundInvalidPost.Success)
   {
      MessageBox.Show("can NOT found the qqPicUrl !");
   }
}
```

参考书目

[1] 【总结】关于（C#和Python中的）正则表达式[1]

[2] perl regex: m//[2]

[3] perl regex: s///[3]

[4] perl regex: qr/STRING/[4]

[5] Perl Regexp-Quote-Like-Operators[5]

[6] [issue14258] Better explain re.LOCALE and re.UNICODE for \S and \W[6]

[7] Regular Expression Options[7]

[8] 【已解决】Perl中的正则表达式的替换和后向引用[8]

[9] ActionScript[9]

[1] http://www.crifan.com/summary_regular_expression_csharp_python/
[2] http://perldoc.perl.org/functions/m.html
[3] http://perldoc.perl.org/functions/s.html
[4] http://perldoc.perl.org/functions/qr.html
[5] http://perldoc.perl.org/perlop.html#Regexp-Quote-Like-Operators
[6] http://python.6.x6.nabble.com/issue14258-Better-explain-re-LOCALE-and-re-UNICODE-for-S-and-W-td4568904.html
[7] http://msdn.microsoft.com/en-us/library/yd1hzczs%28v=vs.71%29.aspx
[8] http://www.crifan.com/perl_regex_replace_backreference/
[9] http://zh.wikipedia.org/wiki/ActionScript