vim 正则表达式 很强大

转自博客http://qianjigui.javaeye.com/blog/368449

毋庸多言，在vim中正则表达式得到了十分广泛的应用。 最常用的 / 和 :s 命令中，正则表达式都是不可或缺的。 下面对vim中的正则表达式的一些难点进行说明。

关于magic

vim中有个magic的设定。设定方法为:

:set magic " 设置magic :set nomagic " 取消magic :h magic " 查看帮助

vim毕竟是个编辑器，正则表达式中包含的大量元字符如果原封不动地引用（像perl 那样）， 势必会给不懂正则表达式的人造成麻烦，比如 /foo(1) 命令， 大多数人都用它来查找foo(1)这个字符串， 但如果按照正则表达式来解释，被查找的对象就成了 foo1 了。

于是，vim就规定，正则表达式的元字符必须用反斜杠进行转义才行， 如上面的例子，如果确实要用正则表达式，就应当写成 /foo\(1\) 。 但是，像 . * 这种极其常用的元字符，都加上反斜杠就太麻烦了。 而且，众口难调，有些人喜欢用正则表达式，有些人不喜欢用……

为了解决这个问题，vim设置了 magic 这个东西。简单地说， magic就是设置哪些元字符要加反斜杠哪些不用加的。 简单来说:

magic (\m)：除了 $. * ^ 之外其他元字符都要加反斜杠。**nomagic** (\M)：除了 $ ^ 之外其他元字符都要加反斜杠。

这个设置也可以在正则表达式中通过 \m \M 开关临时切换。 \m 后面的正则表达式会按照 magic 处理，\M 后面的正则表达式按照 nomagic 处理， 而忽略实际的magic设置。

例如:

/\m.* # 查找任意字符串 /\M.* # 查找字符串 .* （点号后面跟个星号）

另外还有更强大的 \v 和 \V。

\v （即 very magic 之意）：任何元字符都不用加反斜杠**\V** （即 very nomagic 之意）：任何元字符都必须加反斜杠

例如:

/\v(a.c){3}$ # 查找行尾的abcaccadc /\m(a.c){3}$ # 查找行尾的(abc){3} /\M(a.c){3}$ # 查找行尾的(a.c){3} /\V(a.c){3}$ # 查找任意位置的(a.c){3}$

默认设置是 magic，vim也推荐大家都使用magic的设置，在有特殊需要时，直接通过 \v\m\M\V 即可。

本文下面使用的元字符都是 magic 模式下的。

量词

vim的量词与perl相比一点也不逊色。

vim	Perl	意义
*	*	0个或多个(匹配优先)
\+	+	1个或多个(匹配优先)
\? 或 \=	?	0个或1个(匹配优先)，\?不能在 ? 命令（逆向查找）中使用
\{n,m}	{n,m}	n个到m个(匹配优先)
\{n,}	{n,}	最少n个(匹配优先)
\{,m}	{,m}	最多m个(匹配优先)

\{n}	{n}	恰好n个
\{-n,m}	{n,m}?	n个到m个(忽略优先)
\{-}	*?	0个或多个(忽略优先)
\{-1,}	+?	1个或多个(忽略优先)
\{-,1}	??	0个或1个(忽略优先)

从上表中可见，vim的忽略优先量词不像perl的 *? +? ?? 那样，而是统一使用 \{- 实现的。 这大概跟忽略优先量词不常用有关吧。

环视和固化分组

vim居然还支持环视和固化分组的功能，强大，赞一个 关于环视的解释请参考Yurii的《精通正则表达式》一书吧。

vim	Perl	意义
\@=	(?=	顺序环视
\@!	(?!	顺序否定环视
\@<=	(?<=	逆序环视
\@<!	(?<!	逆序否定环视
\@>	(?>	固化分组
\%(atom\)	(?:	非捕获型括号

和perl稍有不同的是，vim中的环视和固化分组的模式的位置与perl不同。 例如，查找紧跟在 foo 之后的 bar，perl将模式写在环视的括号内， 而vim将模式写在环视的元字符之前。

Perl的写法 /(?<=foo)bar/ # vim的写法 /\(foo\)\@<=barvim正则表达式 写道

元字符 说明
. 匹配任意一个字符
[abc] 匹配方括号中的任意一个字符。可以使用-表示字符范围，
如[a-z0-9]匹配小写字母和阿拉伯数字。
[^abc] 在方括号内开头使用^符号，表示匹配除方括号中字符之外的任意字符。
\d 匹配阿拉伯数字，等同于[0-9]。
\D 匹配阿拉伯数字之外的任意字符，等同于[^0-9]。
\x 匹配十六进制数字，等同于[0-9A-Fa-f]。
\X 匹配十六进制数字，等同于[^0-9A-Fa-f]。
\w 匹配单词字母，等同于[0-9A-Za-z_]。
\W 匹配单词字母之外的任意字符，等同于[^0-9A-Za-z_]。
\t 匹配<TAB>字符。
\s 匹配空白字符，等同于[\t]。
\S 匹配非空白字符，等同于[^ \t]。
\a 所有的字母字符. 等同于[a-zA-Z]
\l 小写字母 [a-z]
\L 非小写字母 [^a-z]
\u 大写字母 [A-Z]
\U 非大写字母 [^A-Z]

表示数量的元字符
元字符 说明
* 匹配0-任意个
\+ 匹配1-任意个
\? 匹配0-1个
\{n,m} 匹配n-m个
\{n} 匹配n个
\{n,} 匹配n-任意个
\{,m} 匹配0-m个
_. 匹配包含换行在内的所有字符
\{-} 表示前一个字符可出现零次或多次，但在整个正则表达式可以匹配成功的前提下，匹配的字符数越少越好
\= 匹配一个可有可无的项
_s 匹配空格或断行
_[]

元字符 说明
* 匹配 * 字符。
\. 匹配 . 字符。
\/ 匹配 / 字符。
\\ 匹配 \ 字符。
\[匹配 [字符。

表示位置的符号
元字符 说明
$ 匹配行尾
^ 匹配行首
\< 匹配单词词首
\> 匹配单词词尾

替换变量
在正规表达式中使用 \(和 \) 符号括起正规表达式，即可在后面使用\1、\2等变量来访问 \(和 \) 中的内容。

懒惰模式
\{-n,m} 与\{n,m}一样，尽可能少次数地重复
\{-} 匹配它前面的项一次或0次, 尽可能地少
\| "或"操作符
\& 并列

函数式
:s/替换字符串/\=函数式
在函数式中可以使用 submatch(1)、submatch(2) 等来引用 \1、\2 等的内容，而submatch(0)可以引用匹配的整个内容。

与Perl正则表达式的区别 ?
元字符的区别
Vim语法 Perl语法 含义
\+ + 1-任意个
\? ? 0-1个
\{n,m} {n,m} n-m个
\(和\) (和) 分组

例如:
1, 去掉所有的行尾空格：“:%s/\s\+$//”。“%”表示在整个文件范围内进行替换，“\s”表示空白字符（空格和制表符），“\+”对前面的字符匹 配一次或多次（越多越好），“___FCKpd___0rdquo;匹配行尾（使用“___FCKpd___0rdquo;表示单纯的 “___FCKpd___0rdquo;字符）；被替换的内容为空；由于一行最多只需替换一次，不需要特殊标志。这个还是比较简单 的。(/<Space><Tab>)
2,去掉所有的空白行：“:%s/\(\s*\n\)\+/\r/”。这回多了“ \(”、“\)”、“\n”、“\r”和 “*”。“*”代表对前面的字符（此处为“\s”）匹配零次或多次（越多越好；使用“*”表示单纯的“*”字符），“\n”代表换行符，“\r”代表回 车符，“\(”和“\)”对表达式进行分组，使其被视作一个不可分割的整体。因此，这个表达式的完整意义是，把连续的换行符（包含换行符前面可能有的连续 空白字符）替换成为一个单个的换行符。唯一很特殊的地方是，在模式中使用的是“\n”，而被替换的内容中却不能使用“\n”，而只能使用“\r”。原因是 历史造成的，详情如果有兴趣的话可以查看“:help NL-used-for-Nul”。
3,去掉所有的“//”注释：“:%s!\ s*//.*!!”。首先可以注意到，这儿分隔符改用了“!”，原因是在模式或字符串部分使用了“/”字符，不换用其他分隔符的话就得在每次使用“/”字 符本身时写成“\/”，上面的命令得写成“:%s/\s*\/\/.*//”，可读性较低。命令本身倒是相当简单，用过正则表达式的人估计都知道“.”匹 配表示除换行符之外的任何字符吧。
4,去掉所有的“/* */”注释：“:%s!\s*/*_.\{-}*/\s*! !g”。这个略有点复杂了，用到了几个不太常用的 Vim 正则表达式特性。“_.”匹配包含换行在内的所有字符；“\{-}”表示前一个字符可出现零次或多次，但在整个正则表达式可以匹配成功的前提下，匹配的 字符数越少越好；标志“g”表示一行里可以匹配和替换多次。替换的结果是个空格的目的是保证像“int/* space not necessary around comments */main()”这样的表达式在替换之后仍然是合法的。

:g/^\s*$/d 删除只有空白的行
:s/\(\w\+\)\s\+\(\w\+\)/\2\t\1 将 data1 data2 修改为 data2 data1
:%s/\(\w\+\), \(\w\+\)/\2 \1/ 将 Doe, John 修改为 John Doe
:%s/\<id\>/\=line(".") 将各行的 id 字符串替换为行号
:%s/\(^\<\w\+\>\)/\=(line(".")-10) ."". submatch(1)
将每行开头的单词替换为(行号-10).单词的格式,如第11行的word替换成1. word
排序 :/OB/+1,$!sort

分类: 曾经

Penseur
关注 - 1
粉丝 - 24
+加关注

» 下一篇：欧拉道路 欧拉回路 欧拉图 欧拉半图

posted @ 2011-02-25 10:26 Penseur 阅读(18861) 评论(2) 编辑 收藏

评论

#1楼 2013-08-05 14:13 | 爱让一切都对了

很有用，谢谢。其他很多文章都没讲magic设置。

支持(0) 反对(0)

#2楼 2016-08-14 17:16 | 后营马族子弟

@ 爱让一切都对了
朋友,请教一下：_[] 这个是什么意思啊.你没写..刚学所以不太懂..

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:
· 周鸿祎：你把产品做那么好看，为什么还是不成功？
· 苹果在美国涉嫌侵犯专利 被判赔偿730万美元
· Google发布了个“正经”物联网平台，并“拾获”高通抛来的骁龙处理器
· 英国批准了线粒体置换疗法
· Fedora和Ubuntu曝出0day漏洞
» 更多新闻...

最新知识库文章:

· 高质量的工程代码为什么难写
· 循序渐进地代码重构
· 技术的正宗与野路子
· 陈皓：什么是工程师文化？
· 没那么难，谈CSS的设计模式
» 更多知识库文章...

< 2011年2月 >

日	一	二	三	四	五	六
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	**25**	26
27	**28**	1	2	3	4	5
6	7	8	9	10	11	12

随笔分类

图论(2)
曾经(9)