\\ 反斜杠

\t 间隔 ('\')

\n 换行 ('\')

\r 回车 ('\')

\d 数字 等价于[0-9]

\D 非数字 等价于[^0-9]

\s 空白符号 [\t\n\x0B\f\r]

\S 非空白符号 [^\t\n\x0B\f\r]

\w 单独字符 [a-zA-Z\_0-9]

\W 非单独字符 [^a-zA-Z\_0-9]

\f 换页符

\e Escape

\b 一个单词的边界

\B 一个非单词的边界

\G 前一个匹配的结束

^为限制开头

^java     条件限制为以Java为开头字符

$为限制结尾

java$     条件限制为以java为结尾字符

.  条件限制除\n以外任意一个单独字符

java..     条件限制为java后除换行外任意两个字符

加入特定限制条件「[]」

[a-z]     条件限制在小写a to z范围中一个字符

[A-Z]     条件限制在大写A to Z范围中一个字符

[a-zA-Z] 条件限制在小写a to z或大写A to Z范围中一个字符

[0-9]     条件限制在小写0 to 9范围中一个字符

[0-9a-z] 条件限制在小写0 to 9或a to z范围中一个字符

[0-9[a-z]] 条件限制在小写0 to 9或a to z范围中一个字符(交集)

[]中加入^后加再次限制条件「[^]」

[^a-z]     条件限制在非小写a to z范围中一个字符

[^A-Z]     条件限制在非大写A to Z范围中一个字符

[^a-zA-Z] 条件限制在非小写a to z或大写A to Z范围中一个字符

[^0-9]     条件限制在非小写0 to 9范围中一个字符

[^0-9a-z] 条件限制在非小写0 to 9或a to z范围中一个字符

[^0-9[a-z]] 条件限制在非小写0 to 9或a to z范围中一个字符(交集)

在限制条件为特定字符出现0次以上时，可以使用「\*」

J\*     0个以上J

.\*     0个以上任意字符

J.\*D     J与D之间0个以上任意字符

在限制条件为特定字符出现1次以上时，可以使用「+」

J+     1个以上J

.+     1个以上任意字符

J.+D     J与D之间1个以上任意字符

在限制条件为特定字符出现有0或1次以上时，可以使用「?」

JA?     J或者JA出现

限制为连续出现指定次数字符「{a}」

J{2}     JJ

J{3}     JJJ

文字a个以上，并且「{a,}」

J{3,}     JJJ,JJJJ,JJJJJ,???(3次以上J并存)

文字个以上，b个以下「{a,b}」

J{3,5}     JJJ或JJJJ或JJJJJ

两者取一「|」

J|A     J或A

Java|Hello     Java或Hello

「()」中规定一个组合类型

比如，我查询<a href=\"index.html\">index</a>中<a href></a>间的数据，可写作<a.\*href=\".\*\">(.+?)</a>

在使用Pattern.compile函数时，可以加入控制正则表达式的匹配行为的参数：

Pattern Pattern.compile(String regex, int flag)

flag的取值范围如下：

Pattern.CANON\_EQ     当且仅当两个字符的"正规分解(canonical decomposition)"都完全相同的情况下，才认定匹配。比如用了这个标志之后，表达式"a\?"会匹配"?"。默认情况下，不考虑"规 范相等性(canonical equivalence)"。

Pattern.CASE\_INSENSITIVE(?i)     默认情况下，大小写不明感的匹配只适用于US-ASCII字符集。这个标志能让表达式忽略大小写进行匹配。要想对Unicode字符进行大小不明感的匹 配，只要将UNICODE\_CASE与这个标志合起来就行了。

Pattern.COMMENTS(?x)     在这种模式下，匹配时会忽略(正则表达式里的)空格字符(译者注：不是指表达式里的"\\s"，而是指表达式里的空格，tab，回车之类)。注释从#开始，一直到这行结束。可以通过嵌入式的标志来启用Unix行模式。

Pattern.DOTALL(?s)     在这种模式下，表达式'.'可以匹配任意字符，包括表示一行的结束符。默认情况下，表达式'.'不匹配行的结束符。

Pattern.MULTILINE

(?m)     在这种模式下，'^'和'$'分别匹配一行的开始和结束。此外，'^'仍然匹配字符串的开始，'$'也匹配字符串的结束。默认情况下，这两个表达式仅仅匹配字符串的开始和结束。

Pattern.UNICODE\_CASE

(?u)     在这个模式下，如果你还启用了CASE\_INSENSITIVE标志，那么它会对Unicode字符进行大小写不明感的匹配。默认情况下，大小写不敏感的匹配只适用于US-ASCII字符集。

Pattern.UNIX\_LINES(?d)     在这个模式下，只有'\n'才被认作一行的中止，并且与'.'，'^'，以及'$'进行匹配。

抛开空泛的概念，下面写出几个简单的Java正则用例：

◆比如，在字符串包含验证时

//查找以Java开头,任意结尾的字符串

  Pattern pattern = Pattern.compile("^Java.\*");

  Matcher matcher = pattern.matcher("Java不是人");

  boolean b= matcher.matches();

  //当条件满足时，将返回tr，否则返回false

  System.out.println(b);

◆以多条件分割字符串时

Pattern pattern = Pattern.compile("[, |]+");

String[] strs = pattern.split("Java Hello World  Java,Hello,,World|Sun");

for (int i=0;i<strs.length;i++) {

    System.out.println(strs[i]);

}

◆文字替换（首次出现字符）

Pattern pattern = Pattern.compile("正则表达式");

Matcher matcher = pattern.matcher("正则表达式 Hello World,正则表达式 Hello World");

//替换第一个符合正则的数据

System.out.println(matcher.replaceFirst("Java"));

◆文字替换（全部）

Pattern pattern = Pattern.compile("正则表达式");

Matcher matcher = pattern.matcher("正则表达式 Hello World,正则表达式 Hello World");

//替换第一个符合正则的数据

System.out.println(matcher.replaceAll("Java"));

◆文字替换（置换字符）

Pattern pattern = Pattern.compile("正则表达式");

Matcher matcher = pattern.matcher("正则表达式 Hello World,正则表达式 Hello World ");

StringB?r sbr = new StringB?r();

while (matcher.find()) {

    matcher.appendReplacement(sbr, "Java");

}

matcher.appendTail(sbr);

System.out.println(sbr.toString());

◆验证是否为邮箱地址

String str="ceponline@yahoo.com.cn";

Pattern pattern = Pattern.compile("[\\w\\.\\-]+@([\\w\\-]+\\.)+[\\w\\-]+",Pattern.CASE\_INSENSITIVE);

Matcher matcher = pattern.matcher(str);

System.out.println(matcher.matches());

◆去除html标记

Pattern pattern = Pattern.compile("<.+?>", Pattern.DOTALL);

Matcher matcher = pattern.matcher("<a href=\"index.html\">主页</a>");

String string = matcher.replaceAll("");

System.out.println(string);

◆查找html中对应条件字符串

Pattern pattern = Pattern.compile("href=\"(.+?)\"");

Matcher matcher = pattern.matcher("<a href=\"index.html\">主页</a>");

if(matcher.find())

  System.out.println(matcher.group(1));

}

◆截取http://地址

//截取url

Pattern pattern = Pattern.compile("(http://|https://){1}[\\w\\.\\-/:]+");

Matcher matcher = pattern.matcher("dsdsds<http://dsds//gfgffdfd>fdf");

StringB?r b?r = new StringB?r();

while(matcher.find()){

    b?r.append(matcher.group());

    b?r.append("\r\n");

System.out.println(b?r.toString());

}

◆替换指定{}中文字

String str = "Java目前的发展史是由{0}年-{1}年";

String[][] object={new String[]{"\\{0\\}","1995"},new String[]{"\\{1\\}","2007"}};

System.out.println(replace(str,object));

p  
lic static String replace(final String sourceString,Object[] object) {

            String temp=sourceString;

            for(int i=0;i<object.length;i++){

                      String[] result=(String[])object[i];

               Pattern    pattern = Pattern.compile(result[0]);

               Matcher matcher = pattern.matcher(temp);

               temp=matcher.replaceAll(result[1]);

            }

            return temp;

}

◆以正则条件查询指定目录下文件

 //用于缓存文件列表

        private ArrayList files = new ArrayList();

        //用于承载文件路径

        private String \_path;

        //用于承载未合并的正则公式

        private String \_regexp;

        class MyFileFilter implements FileFilter {

              /\*\*

               \* 匹配文件名称

               \*/

              p  
lic boolean accept(File file) {

                try {

                  Pattern pattern = Pattern.compile(\_regexp);

                  Matcher match = pattern.matcher(file.getName());

                  return match.matches();

                } catch (Exception e) {

                  return tr;

                }

              }

            }

        /\*\*

         \* 解析输入流

         \* @param inputs

         \*/

        FilesAnalyze (String path,String regexp){

            getFileName(path,regexp);

        }

        /\*\*

         \* 分析文件名并加入files

         \* @param input

         \*/

        private void getFileName(String path,String regexp) {

            //目录

              \_path=path;

              \_regexp=regexp;

              File directory = new File(\_path);

              File[] filesFile = directory.listFiles(new MyFileFilter());

              if (filesFile == null) return;

              for (int j = 0; j < filesFile.length; j++) {

                files.add(filesFile[j]);

              }

              return;

            }

        /\*\*

         \* 显示输出信息

         \* @param out

         \*/

        p  
lic void print (PrintStream out) {

            Iterator elements = files.iterator();

            while (elements.hasNext()) {

                File file=(File) elements.next();

                    out.println(file.getPath());

            }

        }

        p  
lic static void output(String path,String regexp) {

            FilesAnalyze fileGroup1 = new FilesAnalyze(path,regexp);

            fileGroup1.print(System.out);

        }

        p  
lic static void main (String[] args) {

            output("C:\\","[A-z|.]\*");

        }