Java 正则

字符	含义	等价(Unicode)
\\	反斜杠	\
\t	间隔	\u0009
\n	换行	\u000A
\r	回车	\u000D
\d	数字	[0-9]
\D	非数字	[^0-9]
\s	空白字符	[\t\n\f\r]
\S	非空白字符	[^\t\n\r\f]
\w	单独字符	[a-zA-Z0-9_]
\W	非单独字符	[^a-zA-Z0-9_]
\f	换页符	
\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	

其他符号含义

符号	含义	详述
^	限制开头	^java：条件限制为以 java 开头的字符串
$	限制结尾	Java$：以 Java 结尾
.	除\n 外任一字符	Java..：以 Java 开头，后接任一除\n 外的两个字符
*	0 到多个	J*：0 到多个 J
+	1 到多个	J+：1 个到多个 J
?	0 到 1 个	J?：0 个或 1 个 J
{n}	重复出现 n 次	P{5}：匹配 5 个 P
{n,}	重复出现 n 次以上	P{3,}：匹配 3 个 P 或 3 个 P 以上
{n,m}	重复出现 n 到 m 次	P{3,5}：匹配 3 到 5 个 P
()	规定一个组合类型	(.*)：0 个或者多个除\n 外的任意字符为一组

如下方法

Pattern　Pattern.compile(String regex,int　flag)

其中 int　flag 参数的含义

常量值	含义
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，默认情况不考虑规范相等性
Pattern.CASE_INSENSITIVE	不区分大小写
Pattern.COMMENTS	忽略表达式内的空格，而不是指\s 之类的
Pattern.DOTALL	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	'^'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	如果你还启用了 CASE_INSENSITIVE 标志，那么它会对 Unicode 字符进行大小写不敏感的匹配。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	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

实例

```
/*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 是什么?”);
/*调用 Matcher 的 matchers 方法进行匹配,当条件满足时返回 true，否
则返回 false*/
if(matcher.matches()){
        System.out.println(“匹配成功”);
}
```

```
/*以多条件分割字符串*/
Pattern   pattern = Pattern.compile(“[,|]”);
String [] str = pattern.split(“Hello , world ! I am Paulz | luozhuan”);
for(int i = 0; i < str.length;i++){
        System.out.println(str[i]);
}
```

```
/*文字替换(首次出现的的字符)*/
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 AUSE!”);
//替换第一个符合正则的字符
System.out.println(matcher.replaceFirst(“Java”));
//输出结果为: Java,Hello World!正则表达式,Hello AUSE!
```

```
/*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 AUSE!”);
System.out.println(matcher.replaceAll(“Java”));
//输出结果为: Java,Hello World!Java,Hello AUSE!
```

```
/*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 AUSE!”);
StringBuffer sb = new StringBuffer();
//find():尝试查找与该模式匹配的输入序列的下一个子序列
while(matcher.find()){
```

```
        //appendReplacement():实现非终端追加和替换步骤
        matcher.appendReplacement(sb,”Java”);
}
//appendTail():实现终端追加和替换步骤
matcher.appendTail(sb);
System.out.println(sb.toString());
```

```
/*验证是否为邮箱地址*/
String email = “Paulz@lsjt.com.cn”;
Pattern pattern =
Pattern.compile(“[\\w\\.\\-]+@([\\w\\-]+\\.)+[\\w\\-]+”,Pattern.CASE_INSENSITIVE);
Matcher matcher = pattern.matcher(email);
System.out.println(matcher.matches());
```

```
/*查找 HTML 中对应条件的字符串*/
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));//index.html
}
```

```
/*截取 URL*/
Pattern pattern =
Pattern.compile(“(http://|https://){1}[\\w\\-\\./]+”);
Matcher matcher =
pattern.matcher(“asdf<http://lsjt/paulz/index.jsp>asdf”);
StringBuffer sb = new StringBuffer();
while(matcher.find()){
   sb.append(matcher.group());//返回以前匹配操作所匹配的序列
   //sb.append(“\r\n”);
 }
System.out.println(sb.toString());
//结果: http://lsjt/paulz/index.jsp
```