厦门大学

正则表达式傻瓜式宝典

Atuthor:xmusoftware

Software School Of Xiamen University

Part Ⅰ基础篇

正则表达式基础知识

我们先从简单的开始。假设你要搜索一个包含字符“cat”的字符串，搜索用的正则表达式就是“cat”。如果搜索对大小写不敏感，单词“catalog”、“Catherine”、“sophisticated”都可以匹配。也就是说：

```
正则表达式：cat
匹配：cat，catalog，Catherine，sophisticated
```

1.1 句点符号

假设你在玩英文拼字游戏，想要找出三个字母的单词，而且这些单词必须以“t”字母开头，以“n”字母结束。另外，假设有一本英文字典，你可以用正则表达式搜索它的全部内容。要构造出这个正则表达式，你可以使用一个通配符——句点符号“.”。这样，完整的表达式就是“t.n”，它匹配“tan”、“ten”、“tin”和“ton”，还匹配“t#n”、“tpn”甚至“t n”，还有其他许多无意义的组合。这是因为句点符号匹配所有字符，包括空格、Tab 字符甚至换行符：

```
正则表达式：t.n
匹配：tan，Ten，tin，ton，t n，t#n，tpn，等
```

1.2 方括号符号

为了解决句点符号匹配范围过于广泛这一问题，你可以在方括号（“[]”）里面指定看来有意义的字符。此时，只有方括号里面指定的字符才参与匹配。也就是说，正则表达式“t[aeio]n”只匹配“tan”、“Ten”、“tin”和“ton”。但“Toon”不匹配，因为在方括号之内你只能匹配单个字符：

```
正则表达式：t[aeio]n
匹配：tan，Ten，tin，ton
```

1.3 “或”符号

如果除了上面匹配的所有单词之外，你还想要匹配“toon”，那么，你可以使用“|”操作符。“|”操作符的基本意义就是“或”运算。要匹配“toon”，使用“t(a|e|i|o|oo)n”正则表达式。这里不能使用方扩号，因为方括号只允许匹配单个字符；这里必须使用圆括号“()”。圆括号还可以用来分组，具

体请参见后面介绍。

```
正则表达式：t(a|e|i|o|oo)n
匹配：tan，Ten，tin，ton，toon
```

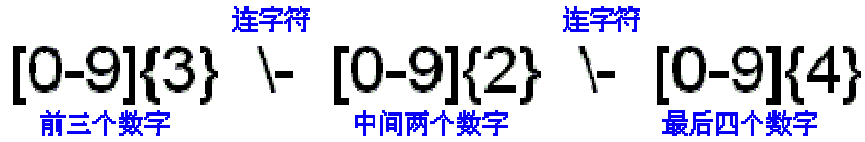
1.4 表示匹配次数的符号

表一显示了表示匹配次数的符号，这些符号用来确定紧靠该符号左边的符号出现的次数：

表一：表示次数的符号

符号	次数
*	0 次或者多次
+	1 次或者多次
?	0 次或者 1 次
{n}	恰好 n 次
{n,m}	从 n 次到 m 次

假设我们要在文本文件中搜索美国的社会安全号码。这个号码的格式是 999-99-9999。用来匹配它的正则表达式如图一所示。在正则表达式中，连字符（“-”）有着特殊的意义，它表示一个范围，比如从 0 到 9。因此，匹配社会安全号码中的连字符号时，它的前面要加上一个转义字符“\”。

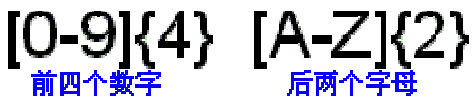


图一：匹配所有 123-12-1234 形式的社会安全号码

假设进行搜索的时候，你希望连字符号可以出现，也可以不出现——即，999-99-9999 和 999999999 都属于正确的格式。这时，你可以在连字符号后面加上“？”数量限定符号，如图二所示：

图二：匹配所有 123-12-1234 和 123121234 形式的社会安全号码

下面我们再来看另外一个例子。美国汽车牌照的一种格式是四个数字加上二个字母。它的正则表达式前面是数字部分“[0-9]{4}”，再加上字母部分“[A-Z]{2}”。图三显示了完整的正则表达式。



图三：匹配典型的美国汽车牌照号码，如 8836KV

1.5 “否”符号

“^”符号称为“否”符号。如果用在方括号内，“^”表示不想要匹配的字符。例如，图四的正则表达式匹配所有单词，但以“X”字母开头的单词除外。

图四：匹配所有单词，但“X”开头的除外

1.6 圆括号和空白符号

假设要从格式为“June 26, 1951”的生日日期中提取出月份部分，用来匹配该日期的正则表达式可以如图五所示：

图五：匹配所有 Moth DD,YYYY 格式的日期

新出现的“\s”符号是空白符号，匹配所有的空白字符，包括 Tab 字符。如果字符串正确匹配，接下来如何提取出月份部分呢？只需在月份周围加上一个圆括号创建一个组，然后用 ORO API（本文后面详细讨论）提取出它的值。修改后的正则表达式如图六所示：

图六：匹配所有 Month DD,YYYY 格式的日期，定义月份值为第一个组

1.7 其它符号

为简便起见，你可以使用一些为常见正则表达式创建的快捷符号。如表二所示：

表二：常用符号

表二：常用符号	
符号	等价的正则表达式
\d	[0-9]
\D	[^0-9]
\w	[A-Z0-9]
\W	[^A-Z0-9]
\s	[\t\n\r\f]
\S	[^ \t\n\r\f]

例如，在前面社会安全号码的例子中，所有出现“[0-9]”的地方我们都可以使用“\d”。修改后的正则表达式如图七所示：

图七：匹配所有 123-12-1234 格式的社会安全号码

Part Ⅱ 应用篇

1. 邮政编码

```
boolean checkPostcode(){
Pattern p=Pattern.compile("[0-9]{6}");
  Matcher m=p.matcher(inputStr);
  if (!m.matches()){
   System.out.println("****邮政编码格式不符！*****");
   return false;
  }
  return true;
 }
```

java.util.regex 中有两个类：Pattern 和 Matcher。

Pattern 为模板，Matcher 为被匹配者。

打个比方，你心中已经有了你将来的另一半将会是怎样的想法（只是打个比方，无意招惹有了另一半的同志们），这个想法就是模板 Pattern，现在你遇见了一位，她就是 Matcher，于是你会用你的 Pattern 去和她比较（matches（））， 如果匹配成功，那么就有可能有一段佳话。

关于上面的程序不做详细解释，可以参考 Java 的 API 中 Pattern 和 Matcher 类的说明。

只介绍一下“[0-9]{6}”的意思。

[]中括号指定允许匹配的字符，恰如您心目中她要：温柔，端庄，聪慧，但是一个[]只能允许匹配单个字符。0-9 表示 0 到 9 之间的任意数字，同样 A-Z 就表示 A 到 Z 之间的任意字母，连接符-的意思相信你已经掌握了。{}大括号表示匹配次数，这里就表示匹配 6 次，即必须有 6 个数字。

2.EMAIL

```
boolean checkEmail(){

  Pattern p=Pattern.compile("[0-9A-Za-z]+@([0-9a-zA-Z]+.){1,2}(com|net|cn|com.cn)");

 Matcher m=p.matcher(inputStr);
  if(!m.matches()){
   System.out.println("****电子邮件格式不符！*****");
   return false;
  }
  return true;
 }
```

这里就不多做解释，之给出匹配的表达式：

“[0-9A-Za-z]+@([0-9a-zA-Z]+.){1,2}(com|net|cn|com.cn)”。

4. IP 地 址

这里用正则表达式我检查指定的字符串是否式一个 IP 地址,注意这里前缀 0 是不允许的,如果允许前缀 0 的 话那问题就简单了好多,这个实现效率比较低下.

boolean ipValid(String s)

```
{

String regex0="(2[0-4]\\d)" + "|(25[0-5])";

String regex1="1\\d{2}";

String regex2="[1-9]\\d";

String regex3="\\d";

String regex="("+regex0+")|("+regex1+")|("+regex2+")|("+regex3+")";
regex="("+regex+").("+regex+").("+regex+").("+regex+")";

Pattern p=Pattern.compile(regex);

Matcher m=p.matcher(s); return m.matches();

}
```

5.正则表达式几种常用功能——查询，提取，替换，分割

正则表达式在字符串处理上有着强大的功能，sun 在 jdk1.4 加入了对它的支持 下面简单的说下它的 4 种常用功能：

查询：

```
String str="abc efg ABC"; String regEx="a|f"; //表示 a 或 f Pattern p=Pattern.compile(regEx);
Matcher m=p.matcher(str); boolean rs=m.find();
```

如果 str 中有 regEx，那么 rs 为 true，否则为 flase。

如果想在查找时忽略大小写，则可以写成 Pattern p=Pattern.compile(regEx,Pattern.CASE_INSENSITIVE);

提取

```
String regEx=".+\\\\(.+)$"; String str="c:\\dir1\\dir2\\name.txt";
Pattern p=Pattern.compile(regEx);
Matcher m=p.matcher(str);
boolean rs=m.find();
for(int i=1;i<=m.groupCount();i++)
{
System.out.println(m.group(i));
}
```

以上的执行结果为 name.txt，提取的字符串储存在 m.group(i)中，其中 i 最大值为 m.groupCount();

分割：

```
String regEx="::";
Pattern p=Pattern.compile(regEx);
String[] r=p.split("xd::abc::cde");
```

执行后，r 就是{"xd","abc","cde"}，其实分割时还有跟简单的方法：String str="xd::abc::cde"; String[]

r=str.split("::");
替换（删除）：
String regEx="a+"; //表示一个或多个 a
Pattern p=Pattern.compile(regEx);
Matcher m=p.matcher("aaabbced a ccdeaa");
String s=m.replaceAll("A");
结果为"Abbced A ccdeA" 如果写成空串，既可达到删除的功能，
比如： String s=m.replaceAll(""); 结果为"bbced ccde"
附：
\d 等於 [0-9] 数字
\D 等於 [^0-9] 非数字
\s 等於 [\t\n\x0B\f\r] 空白字元
\S 等於 [^ \t\n\x0B\f\r] 非空白字元
\w 等於 [a-zA-Z_0-9] 数字或是英文字
\W 等於 [^a-zA-Z_0-9] 非数字与英文字
^ 表示每行的开头
$ 表示每行的结尾

Part Ⅲ 其他

正则表达式用于字符串处理，表单验证等场合，实用高效，但用到时总是不太把握，以致往往要上网查一番。我将一些常用的表达式收藏在这里，作备忘之用。

匹配中文字符的正则表达式： [\u4e00-\u9fa5]

匹配双字节字符(包括汉字在内)：[^\x00-\xff]

应用：计算字符串的长度（一个双字节字符长度计 2，ASCII字符计 1）

```
String.prototype.len=function(){return this.replace([^\x00-\xff]/g,"aa").length;}
```

匹配空行的正则表达式：\n[\s|]*\r

匹配HTML标记的正则表达式：/<(.*)>.*<\/\1>|<(.*) \/>/

匹配首尾空格的正则表达式：(^\s*)|(\s*$)

应用：javascript中没有像vbscript那样的trim函数，我们就可以利用这个表达式来实现，如下：

```
String.prototype.trim = function()
{
return this.replace(/(^\s*)|(\s*$)/g, "");
}
```

利用正则表达式分解和转换IP地址：

下面是利用正则表达式匹配IP地址，并将IP地址转换成对应数值的Javascript程序：

```
function IP2V(ip)
{
re=/(\d+)\.(\d+)\.(\d+)\.(\d+)/g //匹配IP地址的正则表达式
if(re.test(ip))
{
return
RegExp.$1*Math.pow(255,3))+RegExp.$2*Math.pow(255,2))+RegExp.$3*255+R
egExp.$4*1
}
else
{
throw new Error("Not a valid IP address!")
}
}
```

不过上面的程序如果不用正则表达式，而直接用split函数来分解可能更简单，程序如下：

```
var ip="10.100.20.168"
ip=ip.split(".")
alert("IP值是："+(ip[0]*255*255*255+ip[1]*255*255+ip[2]*255+ip[3]*1))
```

匹配Email地址的正则表达式：\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*

匹配网址URL的正则表达式：http://([\w-]+\.)+[\w-]+(/[\w- ./?%&=]*)?

利用正则表达式去除字串中重复的字符的算法程序：

```
var s="abacabefgeeii"
var s1=s.replace(/(.).*\1/g,"$1")
var re=new RegExp("["+s1+"]","g")
var s2=s.replace(re,"")
alert(s1+s2) //结果为：abcefgi
```

我原来在CSDN上发贴寻求一个表达式来实现去除重复字符的方法，最终没有找到，这是我能想到的最简单的实现方法。思路是使用后向引用取出包括重复的字符，再以重复的字符建立第二个表达式，取到不重复的字符，两者串连。这个方法对于字符顺序有要求的字符串可能不适用。

得用正则表达式从URL地址中提取文件名的javascript程序，如下结果为page1

```
s="http://www.9499.net/page1.htm"
s=s.replace(/(.*\/){0,}([^\.]+).*/ig,"$2")
```

alert(s)

利用正则表达式限制网页表单里的文本框输入内容：

用正则表达式限制只能输入中文：onkeyup="value=value.replace(/[^\u4E00-\u9FA5]/g,'')" onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^\u4E00-\u9FA5]/g,''))"

用正则表达式限制只能输入全角字符：onkeyup="value=value.replace(/[^\uFF00-\uFFFF]/g,'')" onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^\uFF00-\uFFFF]/g,''))"

用正则表达式限制只能输入数字：onkeyup="value=value.replace(/[^\d]/g,'')"onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^\d]/g,''))"

用正则表达式限制只能输入数字和英文：onkeyup="value=value.replace(/[\W]/g,'')"onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^\d]/g,''))"