正则表达式专题

讲授：李钦坤

目录

- 正则表达式概述
- Java中的相关API
- 正则表达式语法

正则表达式

- 正则表达式(regular expression)描述了一种字符串匹配的模式。
  - 正则表达式作为一个模板，与所搜索的字符串进行匹配。
  - 常用于：字符串的匹配、查找、替换、分割等

1 正则表达式相关

- java.util.regex包中提供了相关类
  - Pattern：模式类。正则表达式的编译表示形式
  - Matcher：匹配器。用于匹配字符序列与正则表达式模式的类
  - PatternSyntaxException：正则表达式模式中的语法错误
- String类中与正则相关的方法
  - public boolean matches(String regex); 告知此字符串是否匹配给定的正则表达式
  - String[] split(String regex); 根据给定正则表达式的匹配拆分此字符串
  - String replaceAll(String regex, String replacement);使用给定的replacement替换此字符串所有匹配给定的正则表达式的子字符串

Pattern类常用方法

- public static Pattern compile(String regex)
  - 将给定的正则表达式编译到模式中
- public static Pattern compile(String regex, int flags)
  - 将给定的正则表达式编译到具有给定标志的模式中
  - flags参数表示匹配时的选项，常用的flags参数值有：
    - CASE_INSENSITIVE：启用不区分大小写的匹配。
    - COMMENTS：模式中允许空白和注释。
    - MULTILINE：启用多行模式。
- public Matcher matcher(CharSequence input)
  - 生成一个给定命名的Matcher对象
- static boolean matches(String regex, CharSequence input)
  - 直接判断字符序列input是否匹配正则表达式regex。
  - 该方法适合于该正则表达式只会使用一次的情况

Matcher类常用方法

- public boolean matches()
  - 尝试将整个输入序列与该模式匹配。 (开头到结尾)
- public boolean find()
  - 扫描输入序列以查找与该模式匹配的下一个子序列
- public String replaceAll(String replacement)
  - 替换模式与给定替换字符串相匹配的输入序列的每个子序列
- public String replaceFirst(String replacement)
  - 替换模式与给定替换字符串匹配的输入序列的第一个子序列

- public String group()
  - 返回由以前匹配操作所匹配的输入子序列
- public String group(int group)
  - 返回在以前匹配操作期间由给定组捕获的输入子序列

正则表达式语法

- 正则表达式的模式串：是由普通字符（如字符a到z）以及一些特殊字符（称为元字符）组成
- 元字符从功能上分为：限定符、选择匹配符、特殊字符、字符匹配符、定位符、分组组合符、反向引用符。

元字符—限定符

- 1. 限定符：用于指定其前面的单个字符或组合项连续出现多少次

字符	说明
*	匹配前面的字符或子表达式零次或多次。 例如："ja*"匹配"j"和"jaaaaaaaa"。
+	匹配前面的字符或子表达式一次或多次。 例如："ja+"与"ja"和"jaaaaa"匹配，但与"j"不匹配。
?	匹配前面的字符或子表达式零次或一次。 例如："core?"匹配"cor"或"core"。不能匹配"coreeeee"
{n}	正好匹配n次。n是一个非负整数。 例如："o{2}"与"Bob"中的"o"不匹配，与"food"中的两个"o"匹配。
{n,}	至少匹配n次。 例如："o{2,}"与"Bob"中的"o"不匹配，与"foooood"中的所有"o"匹配。
{n,m}	最少匹配n次，且最多匹配m次。m和n均为非负整数，其中n<=m。 例如："o{1,3}"匹配"fooooood"中的头三个 o。注意：不能将空格插入逗号和数字之间。

贪婪匹配 &非贪婪匹配

- 限定符默认都是贪婪匹配，即尽可能多的去匹配字符。
- 当“?”紧跟在任何一个其他限定符(*，+，?，{n}，{n,}，{n,m})后面时，就是非贪婪匹配模式。

元字符—选择匹配符

- 2.选择匹配符："|"，它用于选择匹配两个选项之中的任意一个。
  - 例如：“j|qava” 匹配“j”或“qava”或“java”，"(j|q)ava"匹配"java"或"qava"。

元字符—特殊字符

- 普通字符可以直接用来表示它们本身，也可以用它们的ASCII码或Unicode代码来代表。
  - ASCII码：两位的十六进制值，前面加"\x"
    - 字符b的ASCII码为98(十六进制是62)。所以表示b字符可用"\x62"
  - Unicode码：四位十六进制值，前面加"\u"
    - 匹配字符b，可以用它的Unicode代码"\u0062"
    - 最常用情况是用来表示中文字符的范围："\u4E00-\u9FA5"
- 元字符中用到的特别字符，当作普通字符使用时需要用"\"进行转义：
  - 使用[]来引用也可以当作普通字符来使用
  - "*"匹配"*"字符、"\?"匹配问号字符、
  - "\n"匹配换行符、"\r"匹配回车符、
  - "\t"匹配制表符、\v 匹配垂直制表符、 "\f"匹配换页符

元字符—字符匹配符

- 用于匹配指定字符集合中的任意一个字符

字符	说明
[...]	字符集。匹配指定字符集合包含的任意一个字符。 例如："[abc]"可以与"a"、"b"、"c"三个字符中的任何一个匹配。
[^...]	反向字符集。匹配指定字符集合未包含的任意一个字符。 例如："[^abc]"匹配"a"、"b"、"c"三个字符以外的任意一个字符。
[a-z]	字符范围。匹配指定范围内的任意一个字符。 例如："[a-z]"匹配"a"到"z"范围内的任何一个小写字母。"[0-9]"匹配"0"到"9"范围内容的任何一个数字。
[^a-z]	反向字符范围。匹配不在指定范围内的任何一个字符。 例如："[^a-z]"匹配任何不在"a"到"z"范围内的任何一个字符。"[^0-9]"匹配任何不在"0"到"9"范围内容的任何一个字符。
.	匹配除"\n"之外的任何单个字符。若要匹配包括"\n"在内的任意字符，使用如"[\s\S]"之类的模式。
\d	数字字符匹配。等效于[0-9]。
\D	非数字字符匹配。等效于[^0-9]。
\w	匹配任何单词字符，包括下划线。与"[A-Za-z0-9_]"等效。
\W	与任何非单词字符匹配。与"[^A-Za-z0-9_]"等效。
\s	匹配任何空白字符，包括空格、制表符、换页符等。与"[\f\n\r\t\v]"等效。
\S	匹配任何非空白字符。与"[^\f\n\r\t\v]"等效。

元字符—定位符

- 用于规定匹配模式在目标字符串中的出现位置

字符	说明
^	匹配输入字符串的开始位置。^必须出现在正则模式文本的最前才起定位作用。
$	匹配输入字符串的结尾位置。$必须出现在正则模式文本的最面才起定位作用。
\b	匹配一个单词边界。 例如："er\b"匹配"never love"中的"er"，但不匹配"verb"中的"er"。
\B	非单词边界匹配。 例如："er\B"匹配"verb"中的"er"，但不匹配"never"中的"er"。

元字符—分组组合符

- 用()将正则表达式中的某一部分定义为"组"，并且将匹配这个组的字符保存到一个临时区域。

字符	说明
(pattern)	捕获性分组。将圆括号中的pattern部分组合成一个组合项当作子匹配，每个捕获的子匹配项按照它们在正则模式中从左到右出现的顺序存储在缓冲区中，编号从1开始，可供以后使用。 例如：“(dog)\\1"可以匹配"dogdogdog"中的"dogdog"。
(?:pattern)	非捕获性分组。即把pattern部分组合成一个组合项，但不能捕获供以后使用。
(?=pattern)	正向预测匹配分组。 例如："Windows(?=95|98|NT|2000)"能匹配"Windows2000"中的"Windows"，但不能匹配"Windows3.1"中的"Windows"。
(?!pattern)	正向否定预测匹配分组
(?<=pattern)	负向预测匹配分组。例如："(?<=95|98|NT|2000)Windows"
(?<!pattern)	负向否定预测匹配分组

元字符—其它

- 反向引用符：用于对捕获分组后的组进行引用的符号。格式为"\组编号"
  - 例如：要匹配"goodgood study, dayday up!"中所有连续重复的单词部分，可使用"\b([a-z]+)\1\b"来匹配。
- "$分组编号"可用来引用分组匹配到的结果字符串

- 非贪婪模式尽可能少的匹配所搜索的字符串
  - 当“?”紧跟在任何一个其他限制符(*,+,?，{n}，{n,}，{n,m})后面时，就是使用非贪婪匹配模式。
  - 例如：对于"oooo"串，"o+?"将匹配单个"o"，而"o+"将匹配所有"o"。

示例

- 验证中文名是否合法
- 验证Email地址
- 验证手机号
- 验证邮政编码
- 验证IPv4的ip地址
- 验证国内电话号码
- 从"c:/abc/bcd/def.txt"中提取出文件名
- 把"我我我要要学java"替换成"我要学java"

```
String str = "我我我要要学java";
Matcher m = Pattern.compile("(.)\\1+").matcher(str);
str = m.replaceAll("$1");
System.out.println(str);
```

总结

- 正则表达式(java.util.regex包)
  - Pattern
  - Matcher
- 正则表达式语法
  - 元字符