**JAVA** 正则表达式实例教程

**Nick Shanghai China** [**Kandk522@hotmail.com**](mailto:Kandk522@hotmail.com)

**2008-12-12**

**So many open source projects. Why not Open your Documents?**

**1**、 正则表达式的知识要点

1、 正则表达式是什么？

正则表达式是一种可以用于模式匹配和替换的强有力的工具。

2、 正则表达式的优点？ 更少的代码，更高的效率。

3、 正则表达式如何使用？ 在下面的教程中我们将学习到正则表达式的用法。

4、 正则表达式的缺点？ 需要花一定的时间学习，这个时间由个人决定，一天或者更久一些。

**2**、 正则表达式的各种符号及其含义

|  |  |
| --- | --- |
|  | |
| 常用的符号 | |
| **.** | 表示任意一个字符 |
| **\s** | 空格字符(空格键, tab, 换行, 换页, 回车) |
| **\S** | 非空格字符([^\s]) |
| **\d** | 一个数字，（相当于[0-9] ） |
| **\D** | 一个非数字的字符，（相当于[^0-9] ） |
| **\w** | 一个单词字符(word character) （相当于 [a-zA-Z\_0-9] ） |
| **\W** | 一个非单词的字符，[^\w] |
| **^** | 一行的开始 |
| **$** | 一行的结尾 |
| **\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 |
| **[]** | 匹配方括号内的一个字符 |
| 例如：[abc] 表示字符 a，b，c 中的任意一个(与 a|b|c 相同)  [a-zA-Z] 表示从 a 到 z 或 A 到 Z 当中的任意一个字符 |
| 表示次数的符号 | |
| **\*** | 重复零次或更多次 |
| 例如：a\* 匹配零个或者多个 a |
| **+** | 重复一次或更多次 |
| 例如：a+ 匹配一个或者多个 a |
| **?** | 重复零次或一次 |
| 例如：a? 匹配零个或一个 a |
| **{n}** | 重复 n 次 |
| 例如：a{4} 匹配 4 个 a |
| **{n,}** | 重复 n 次或更多次 |
| 例如：a{4,} 匹配至少 4 个 a |
| **{n,m}** | 重复 n 到 m 次 |
| 例如：a{4,10} 匹配 4~10 个 a |

**3**、 正则表达式实例

java 正则表达式通过 java.util.regex 包下的 Pattern 类与 Matcher 类实现：

1、首先引入 java 包：java.util.regex

2、使用方法如下：

共分三步：

①构造一个模式.

Pattern p=Pattern.compile("[a-z]\*");

②建造一个匹配器

Matcher m = p.matcher(str);

③进行判断，得到结果

boolean b = m.matches()；

Matcher 类提供三个匹配操作方法,三个方法均返回 boolean 类型,当匹配到 时返回 true,没匹配到则返回 false

① m.matches()

matches()对整个字符串进行匹配,只有整个字符串都匹配了才返回 true

② m.lookingAt() lookingAt()对前面的字符串进行匹配,只有匹配到的字符串在最前面才返 回 true

③ m.find()

find()对字符串进行匹配,匹配到的字符串可以在任何位置

Matcher 类的其他方法

int groupcount() 返回此匹配器模式中的捕获组数。

String replaceAll(String replacement) 用给定的 replacement 全部替代匹配的 部分

String repalceFirst(String replacement) 用给定的 replacement 替代第一次匹 配的部分

appendReplacement(StringBuffer sb,String replacement) 根 据 模 式 用 replacement 替换相应内容,并将匹配的结果添加到 sb 当前位置之后 StringBuffer appendTail(StringBuffer sb) 将输入序列中匹配之后的末尾字串 添加到 sb 当前位置之后.

group(n) 0 代表永远都是匹配整个表达式的字符串的那部分 n<>0 时代表 第 n 组匹配的部分

① 字符匹配

Pattern p = Pattern.compile(expression); // 正则表达式

Matcher m = p.matcher(str); // 操作的字符串

boolean b = m.matches(); //返回是否匹配的结果

System.out.println(b);

Pattern p = Pattern.compile(expression); // 正则表达式

Matcher m = p.matcher(str); // 操作的字符串

boolean b = m. lookingAt (); //返回是否匹配的结果

System.out.println(b);

Pattern p = Pattern.compile(expression); // 正则表达式

Matcher m = p.matcher(str); // 操作的字符串

boolean b = m..find (); //返回是否匹配的结果

System.out.println(b);

② 分割字符串

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

String[] strs = pattern.split(str); //操作字符串 得到返回的字符串数组

③ 替换字符串

Pattern p = Pattern.compile(expression); // 正则表达式

Matcher m = p.matcher(text); // 操作的字符串

String s = m.replaceAll(str); //替换后的字符串

④ 查找替换指定字符串

Pattern p = Pattern.compile(expression); // 正则表达式

Matcher m = p.matcher(text); // 操作的字符串

StringBuffer sb = new StringBuffer();

int i = 0;

while (m.find()) {

m.appendReplacement(sb, str);

i++; //字符串出现次数

}

m.appendTail(sb);//从截取点将后面的字符串接上

String s = sb.toString();

⑤ 查找输出字符串

Pattern p = Pattern.compile(expression); // 正则表达式

Matcher m = p.matcher(text); // 操作的字符串

while (m.find()) {

//m.start() 返回匹配到的子字符串在字符串中的索引位置.

//m.end()返回匹配到的子字符串的最后一个字符在字符串中的索引位置.

//m.group()返回匹配到的子字符串

}

3、下面通过几个例子来理解一下正则表达式的妙用

3.1 匹配字符串(matches()方法)

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest{

**static void** test(){

Pattern p = **null**; //正则表达式 Matcher m = **null**; //操作的字符串 **boolean** b = **false**;

//正则表达式表示首字母是a，中间是任意字符，结尾以b结束

//不匹配的结果

p = Pattern.*compile*("a\*b");

m = p.matcher("baaaaab");

b = m.matches(); System.*out*.println("匹配结果："+b); //输出：true

//匹配的结果

p = Pattern.*compile*("a\*b");

m = p.matcher("aaaaab");

b = m.matches(); System.*out*.println("匹配结果："+b); //输出：false

}

**public static void** main(String argus[]){

*test*();

}

}

3.2 判断手机号码(matches()方法）

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest{

**static void** test(){

Pattern p = **null**; //正则表达式 Matcher m = **null**; //操作的字符串 **boolean** b = **false**;

//正则表达式表示第一位是1，第二位为3或5，结尾为9位数字的一串数字

p = Pattern.*compile*("^[1][3,5]+\\d{9}");

m = p.matcher("13812345678");

b = m.matches(); System.*out*.println("手机号码正确："+b); //输出：true

//

p = Pattern.*compile*("[1][3,5]+\\d{9}");

m = p.matcher("03812345678");//错误 首位为0

//m = p.matcher("13812345-7a");//错误 字符串中有字母或者字符

b = m.matches();

System.*out*.println("手机号码错误："+b); //输出：false

}

**public static void** main(String argus[]){

*test*();

}

}

3.3 身份证号码验证

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest{

**static void** test(){

Pattern p = **null**; //正则表达式 Matcher m = **null**; //操作的字符串 **boolean** b = **false**;

//正则表达式表示15位或者18位数字的一串数字

p = Pattern.*compile*("\\d{15}|\\d{18}");

m = p.matcher("120101198506020080");

b = m.matches(); System.*out*.println("身份证号码正确："+b); //输出：true

//

p = Pattern.*compile*("\\d{15}|\\d{18}");

m = p.matcher("020101198506020080");//错误 首位为0 b = m.matches();

System.*out*.println("身份证号码错误："+b); //输出：false

}

**public static void** main(String argus[]){

*test*();

}

}

3.4 email 验证

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest{

**static void** test(){

Pattern p = **null**; //正则表达式 Matcher m = **null**; //操作的字符串 **boolean** b = **false**;

//正则表达式表示邮箱号码

p = Pattern.*compile*("\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+ ([-.]\\w+)\*");

m = p.matcher(["user@test.com"](mailto:user@test.com));

b = m.matches(); System.*out*.println("email号码正确："+b); //输出：true

//

p = Pattern.*compile*("\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+

([-.]\\w+)\*");

m = p.matcher("user.test.com");//错误 @应为. b = m.matches();

System.*out*.println("email号码错误："+b); //输出：false

}

**public static void** main(String argus[]){

*test*();

}

}

3.5 IP 地址检查

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest{

**static void** test(){

Pattern p = **null**; //正则表达式 Matcher m = **null**; //操作的字符串 **boolean** b = **false**;

//正则表达式表示邮箱号码

p = Pattern.*compile*("\\d{1,3}+\\.\\d{1,3}+\\.\\d{1,3}+\\.\\d{1,3}");

m = p.matcher("192.168.1.1");

b = m.matches(); System.*out*.println("IP正确："+b); //输出：true

//

p =

Pattern.*compile*("\\d{1,3}+\\.\\d{1,3}+\\.\\d{1,3}+\\.\\d{1,3}");

m = p.matcher("192.168.1.1234");//错误 应该为3位不应该是4位

b = m.matches(); System.*out*.println("IP错误："+b); //输出：false

}

**public static void** main(String argus[]){

*test*();

}

}

3.6 中文名

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest{

**static void** test(){

Pattern p = **null**; //正则表达式 Matcher m = **null**; //操作的字符串 **boolean** b = **false**;

//正则表达式表示汉字的一串字符串

p = Pattern.*compile*("^[\u4e00-\u9fa5]+$");

m = p.matcher("貂禅");

b = m.matches(); System.*out*.println("中文名正确："+b); //输出：true

//

p = Pattern.*compile*("^[\u4e00-\u9fa5]+$");

m = p.matcher("nick");//错误 只能是中文

b = m.matches();

System.*out*.println("中文名错误："+b); //输出：false

}

**public static void** main(String argus[]){

*test*();

}

}

3.7 字符串匹配的返回

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest {

**static void** test() {

/\*\*

\* start()返回匹配到的子字符串在字符串中的索引位置.

\* end()返回匹配到的子字符串的最后一个字符在字符串中的索引位置.

\* group()返回匹配到的子字符串

\*/

Pattern p=Pattern.*compile*("\\d+"); Matcher m=p.matcher("aaa2223bb11222"); **while**(m.find()){

System.*out*.println(m.start());//第一个循环返回3，第二个循环返回

9

System.*out*.println(m.end());//返回7,第二个循环返回14

System.*out*.println(m.group());//返回2233，第二个返回11222

}

}

**public static void** main(String argus[]) {

*test*();

}

}

3.8 groupCount、group()、group(n)的用法

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest {

**static void** test() {

/\*

\* 本例groupCount将返回三组a11bbb、11、bbb

\*/

Pattern p=Pattern.*compile*("\\w(\\d\\d)(\\w+)"); Matcher m=p.matcher("aa11bbb");

**if**(m.find()){

**int** count = m.groupCount(); //返回匹配组的数目，而不是匹配字符串

的数目

}

}

/\*

**for**(**int** i = 0; i <= count; i++)

System.*out*.println("group " + i + " :" + m.group(i));

\* 返回结果如下

\* group 0 :a11bbb

\* group 1 :11

\* group 2 :bbb

\*/

**public static void** main(String argus[]) {

*test*();

}

}

3.9 分割字符串(split ()方法)

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest {

**static void** test() {

Pattern p=Pattern.*compile*("\\d+"); //将所含数字去掉

String str[] = p.split("aa11bbb33cc55gg");

**for** (**int** i = 0; i < str.length; i++) { System.*out*.println(str[i]);

}

}

/\*

\* 返回结果如下

\* aa

\* bbb

\* cc

\* gg

\*/

**public static void** main(String argus[]) {

*test*();

}

}

3.10 字符串替换(replaceAll()方法)

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest {

**static void** test() {

Pattern p = Pattern.*compile*("\\d+"); //将所含数字全部替换为XX Matcher m = p.matcher("aa11bbb33cc55gg");

String str = m.replaceAll("XX"); System.*out*.println(str);

}

/\*

\* 返回结果如下

\* aaXXbbbXXccXXgg

\*/

**public static void** main(String argus[]) {

*test*();

}

}

3.11 字符串替换 2(appendReplacement()方法及 appendTail()方法)

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public class** RegTest {

**static void** test() {

Pattern p = Pattern.*compile*("55");

Matcher m = p.matcher("aa11bbb33cc55gg55yy"); StringBuffer sb = **new** StringBuffer();

**while** (m.find()) {

m.appendReplacement(sb, "@@");//用@@替换所有的55

} System.*out*.println(sb.toString());//打印aa11bbb33cc@@gg@@ m.appendTail(sb);//将最后一次替换后的字符串加上 System.*out*.println(sb.toString());//打印aa11bbb33cc@@gg@@yy

}

/\*

\* 返回结果如下 aa11bbb33cc@@gg@@、aa11bbb33cc@@gg@@yy

\*/

**public static void** main(String argus[]) {

*test*();

}

}

**4**、 常用的正则表达式

//email 正确的书写格式为 username@domain

**static** String *\_email* = "\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+ ([-.]\\w+)\*";

//电话 正确格式 012-87654321、0123-87654321、0123－7654321

**static** String *\_tel* =

"\\d{4}-\\d{8}|\\d{4}-\\d{7}|\\d(3)-\\d(8)";

//手机号码 13187654321 13587654321

**static** String *\_phone* = "^[1][3,5]+\\d{9}";

//汉字 中国

**static** String *\_chinse* = "^[\u4e00-\u9fa5]+$";

//IP 192.168.1.1

**static** String *\_ip* = "\\d{1,3}+\\.\\d{1,3}+\\.\\d{1,3}+\\.\\d{1,3}";

//QQ 456789

**static** String *\_qq* = "[1-9][0-9]{4,13}";

//邮编 210000

**static** String *\_mail* = "[1-9]\\d{5}(?!\\d)";

//身份证号码 15或者18位

**static** String *\_id* = "\\d{15}|\\d{18}";

**5**、 结尾

1、通过以上的学习，大家可能对正则表达式有了初步的了解和认识，大家在以后的学 习和工作中逐渐加深对 Java 正则表达式的理解，一定会在编程的过程中起到很大作用。