正则表达式

正则表达式是一个强大的字符串处理工具，可以对字符串进行查找，提取，分割，替换等操作。实际上就是自定义一种规则让字符串进行匹配(比如汉语中的AABB，AABC等词语的形式），含有这种定义的形式的就可以进行一些运算了。

1. 使用正则表达式

在程序中定义了正则表达式后就可以通过Pattern和Matcher来使用正则表达式了。

Pattern对象是正则表达式编译后在内存中的表示形式，因此，正则表达式字符串必须先被编译成Pattern对象，然后再通过该对象创建一个对应的Matcher对象。执行匹配所涉及的状态保留在Matcher对象中，多个Matcher对象可共享一个Pattern对象。

1. boolean find（）：此方法放回目标字符串是否包含与Pattern匹配的子串；

**package** 正则表达式;

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

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

**public** **class** Test {

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

String str=**new** String("hello,,world!");

//匹配所有的单词字符

Pattern pattern=Pattern.*compile*("\\w\*");

Matcher matcher =pattern.matcher(str);

System.*out*.println(matcher.find());

System.*out*.println(matcher.start());

System.*out*.println(matcher.find());

System.*out*.println(matcher.start());

System.*out*.println(matcher.find());

System.*out*.println(matcher.start());

}

}

输出是：

true

0

true

5

true

6

由此可见此方法的特点：0位置代表目标字符串的首字母位置。除了第一次执行外，每执行一次光标都会移动，若满足正则表达式则移动到下一次不满足的位置，若不满足正则表达式则移动一个位置

1. int start():返回上一次与Pattern对象匹配的子串在目标字符串中的开始位置

如上面的代码，第一次执行find时由于符合正则表达式，光标一次性的移动到了5的位置，但是第一次匹配时字符串的开始位置是0，所以看到输出0；但是第二次光标已经在5的位置，此时进行匹配的子串的开始位置就是5，所以输出5

1. int end():返回上一次与Pattern对象匹配的子串在目标字符串中的结束位置加1
2. String group():返回上一次与Pattern匹配的子串。（查阅api文档可发现，返回的可能为空字符串，也就是当不匹配时返回一个空字符串）

如代码：

public class Test {

public static void main(String[]args){

String str=new String("hello,,world!");

//匹配所有的单词字符

Pattern pattern=Pattern.*compile*("\\w\*");

Matcher matcher =pattern.matcher(str);

while (matcher.find())

{

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

}

}

}

控制台输出结果是：

hello

world

看到明显输出了两次空字符串

1. Boolean lookingAt():返回目标字符串前面部分与Pattern是否匹配。
2. Boolean matches():返回整个目标字符串与Pattern是否匹配

//上面的代码实例中如果加上

System.out.println(matcher.matches());将返回false，此点与find方法不同

1. Reset():将现有的matcher对象应用与一个新的字符序列。