# 正则表达式

matches

1.字符串：”\\w+;” //单词，数字，下划线

2.a到z: “[a-z]”.

3.排除：”[^a-z]”

4.多个个大范围的写法：”[a-zA-Z]”

5.大范围排除一个部分： “[a-z&&[^fjk]]”

6.空白字符：”[\\s](file:///\\s)” //不可见字符

7.任意一个字符：“.”

8.匹配长度：”.{2}” //2为匹配字符长度！

9.匹配长度为2个数之间 “.{2,5}“

10.数字 [\\d](file:///\\d)

11.至少2个”.{2,}”

12.?: “.?” //0个或者1个 //“.{0,1}“ o个或者1个

13.\* “.\*” //{0,} 至少0个

14.+ “.+” //{1,} 至少1个

15.边界匹配：

15.1：”^\\w“ 匹配第一个字母；开始边界

15.2：”\\w&” 匹配最后一个字母；结束边界

16.单词首字母 “\b\w”

17.反向应用:\1 和第一分组相同

18.反向预搜索：(?<=k)前面有一个字母为k的字符

(?<!k)前面有一个字母不为a的字符

19.正向预搜索：(?=d)后面一个字母为d 的字符

(?!d)后面一个字母不为d的字符

20.(?s)在匹配字符时匹配换行符

21.(?sm)着行匹配

22.(?i)不区分大小写 // 三个模式可以混合使用(?smi)

在java中的应用：

小写转大写方法： .toUpperCase()

String string ="dasdad <A>dfdsf</A> fdsf f<c>dsf</c>dasdad <A>dfdsf</A> fdsf f<c>dsf";

Pattern pattern = Pattern.*compile*("<(\\w+)>(.+?)</\\1>");

Matcher matcher =pattern.matcher(string);

**while** (matcher.find()) {

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

}

替换 : StringBuffer sb =**new** StringBuffer();

**while** (matcher.find()) {

matcher.appendReplacement(sb, "哈哈哈哈哈");

}

matcher.appendTail(sb);

System.***out***.println(sb);

替换：string.replaceAll("<(\\w+)>(.+?)</\\1>","哈哈哈哈")

//第一个参数搜索规则，第二个参数替换为什么