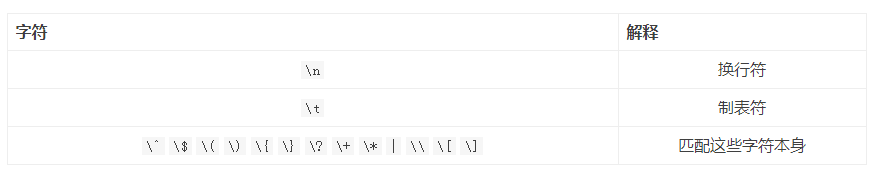
正则语法

普通字符

字母, 数字, 汉字, 下划线, 以及没有特殊定义的标点符号都是普通字符. 表达式中的普通字符在匹配一个字符串时, 匹配与之相同的一个字符.

转义字符



标准字符集合



自定义字符集合



正则表达式中的特殊符号, 被包含到[]中, 则失去特殊意义, 除了- ^之外. 

[]中的^表示取反的含义.

[]中的-表示范围的含义

标准字符集合(除小数点外),如果被包含于中括号中,则自定义集合将包含该集合.   
如[\d.\-+]将匹配: 数字, 小数点, -, +

量词

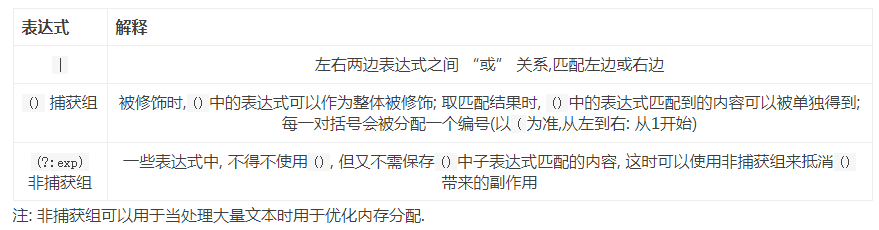


字符边界

零宽匹配: 匹配的不是字符而是位置(符合某种条件的位置),不匹配任何字符.



选择符与分组



零宽断言

零宽度: 只进行子表达式的匹配, 匹配内容不计入最终的匹配结果.

位置匹配: 判断当前位置的前后字符是否符合指定的条件, 但不保留前后的字符.

正则表达式中, 如果子表达式匹配到得是字符内容, 而非位置, 并被保留到最终的匹配结果中, 那么就认为这个子表达式是占有字符的; 如果子表达式匹配的仅仅是位置, 或者匹配的内容并不保存到最终的匹配结果中, 那么就认为这个子表达式是零宽度的(占有字符或零宽度, 是针对匹配的内容是否保留到最终结果而言的)



示例 ：  
 [a-z]+(?=ing) 匹配所有以ing结尾的单词, 但ing并不放入字符串   
 [a-z]+(?=\d+) 匹配所有以数字结尾的单词   
 [a-z]+(?!\d+) 匹配不以数字结尾的单词   
 (?<=(href=\")) 匹配以href="开头的字符串

Pattern 和 Matcher

Pattern对象是正则表达式编译后在内存中的表现形式, 因此, 正则表达式字符串 必须先被编译为Pattern对象Pattern pattern = Pattern.compile("\\w+");然后再利用 该Pattern对象创建对象的Matcher对象Matcher matcher = pattern.matcher(input);

Matcher对象是一个对CharSequence执行匹配操作的正则引擎: 执行匹配所涉及的 状态保留在Matcher对象中, 多个Matcher对象可共享同一个Pattern对象.

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class RegexTest {

public static void main(String[] args) {

String input = "Hello&2018" ;

// 将正则表达式编译成 Pattern 对象

Pattern pattern = Pattern.compile("\\w+"); // 匹配任意一个字母、数字、下划线 "+"号表示一个或以上

Matcher matcher = pattern.matcher(input);

System.out.println(matcher.matches()); // "Hello\_2018" -> true | "Hello&2018" -> false

matcher.reset();

while(matcher.find()){

String group = matcher.group();

System.out.println(group); // Hello 2018

}

String input2 = "123yyy1234" ;

Pattern pattern2 = Pattern.compile("\\d"); // 匹配单个数字

Matcher matcher2 = pattern2.matcher(input2);

String afterReplaceStr = matcher2.replaceAll("#"); // 将匹配到的数字全部换为 "#"

System.out.println(afterReplaceStr); // ###yyy####

String input3 = "1j24h356hijack467g589o" ;

String[] split = input3.split("\\d+"); // 以单个或者多个数字来分割

for(String str : split){

System.out.println(str);

}

}

}