

正则表达式

及**java**文本复杂操作

讲师：高淇

正则表达式课程规划

- 正则表达式基本知识：
 - 基本语法
 - 高级语法
 - 练习
 - editplus,notpad++,ultraedit,eclipse中使用正则
- JAVA复杂文本操作

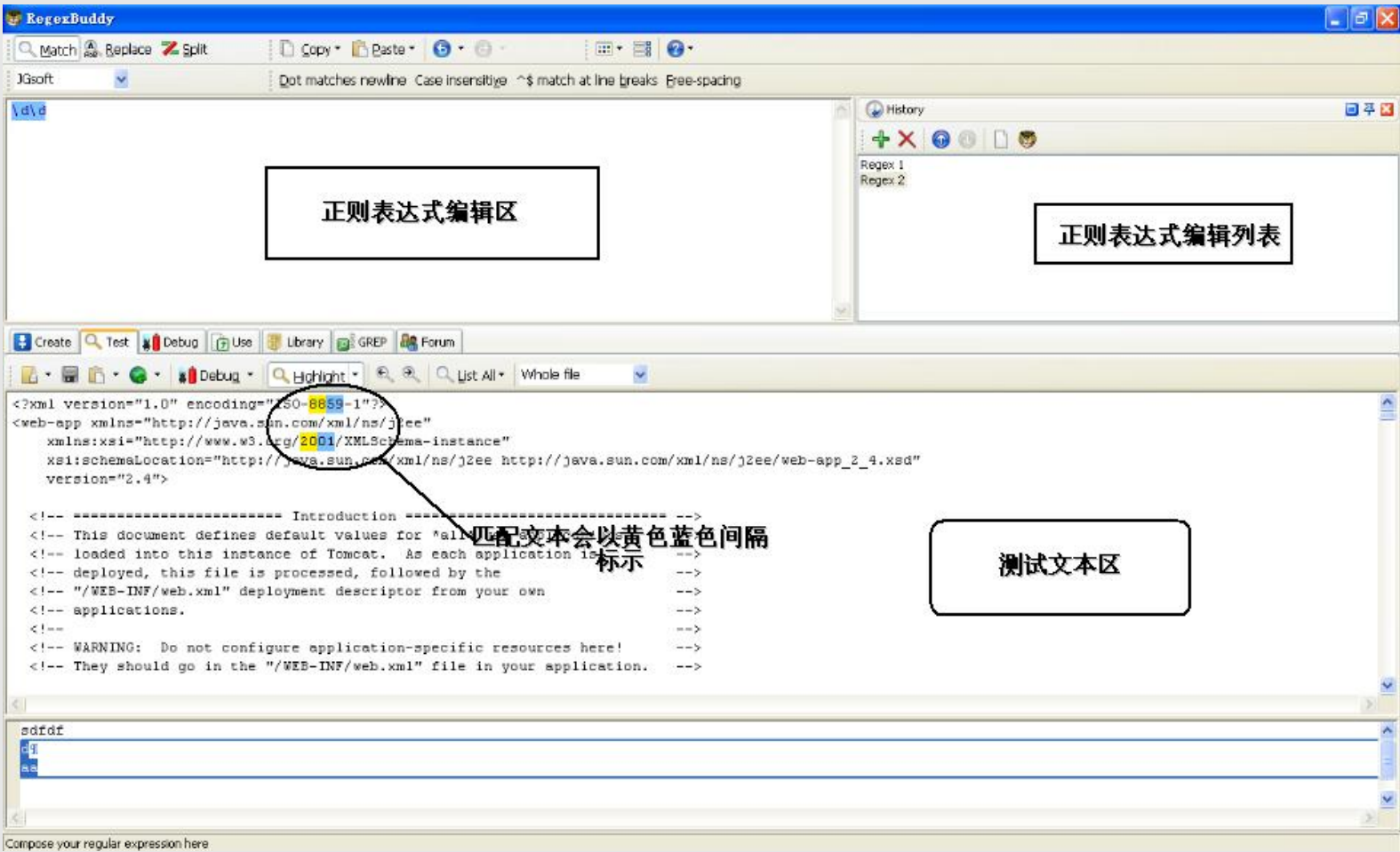
正则表达式(Regular Expresssion)简介

- 为什么需要正则表达式？
 - **文本的复杂处理。**
- 正则表达式的优势和用途？
 - 一种强大而灵活的文本处理工具；
 - 大部分编程语言、数据库、文本编辑器、开发环境都支持正则表达式。
- 正则表达式定义：
 - 正如他的名字一样是描述了一个规则，通过这个规则可以匹配一类字符串。
 - 学习正则表达式很大程度上就是学习正则表达式的**语法规则**。

开发中如何使用？

- 开发中使用正则表达式的流程：
 - 分析所要匹配的数据，写出测试用的典型数据
 - 在工具软件中进行匹配测试
 - 在程序中调用通过测试的正则表达式

工具软件RegexBuddy



正则表达式语法（1）

- 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，在匹配一个字符串的时候，**匹配与之相同的一个字符**。

- 简单的转义字符

| | |
|--|----------|
| \n | 代表换行符 |
| \t | 制表符 |
| \\ | 代表\本身 |
| \^, \\$, \., \[, \], \{, \}, \?, \+, *, \\, \[, \] | 匹配这些字符本身 |

正则表达式语法（2）

- **标准字符集合：**

- 能够与 ‘多种字符’ 匹配的表达式
- **注意区分大小写，大写是相反的意思**

| | |
|----|---|
| \d | 任意一个数字，0~9 中的任意一个 |
| \w | 任意一个字母或数字或下划线，也就是 A~Z,a~z,0~9,_ 中任意一个 |
| \s | 包括空格、制表符、换行符等空白字符的其中任意一个 |
| . | 小数点可以匹配任意一个字符(除了换行符) 如果要匹配包括 “\n”在内的所有字符，一般用[\s\S] |

正则表达式语法（3）·

- **自定义字符集合：**

- []方括号匹配方式，能够匹配方括号中**任意一个**字符

| | |
|-----------|----------------------------------|
| [ab5@] | 匹配 "a" 或 "b" 或 "5" 或 "@" |
| [^ abc] | 匹配 "a","b","c" 之外 的任意一个字符 |
| [f-k] | 匹配 "f"~"k" 之间的任意一个字母 |
| [^A-F0-3] | 匹配 "A"~"F","0"~"3" 之外的任意一个字符 |

- 正则表达式的特殊符号，被包含到中括号中，则失去特殊意义，除了 ^, - 之外。
- 标准字符集合，除小数点外，如果被包含于中括号，自定义字符集合将包含该集合。比如：
 - [\d.\-+]将匹配：数字、小数点、+、-

正则表达式语法（4）

- **量词（Quantifier）**
 - 修饰匹配次数的特殊符号

| | |
|----------|-----------------------|
| {n} | 表达式重复n次 |
| {m,n} | 表达式至少重复m次，最多重复n次 |
| {m,} | 表达式至少重复m次 |
| ? | 匹配表达式0次或者1次，相当于 {0,1} |
| + | 表达式至少出现1次，相当于 {1,} |
| * | 表达式不出现或出现任意次，相当于 {0,} |

- 匹配次数中的**贪婪模式**(匹配字符越多越好，默认！)
- 匹配次数中的**非贪婪模式**（匹配字符越少越好，修饰匹配次数的特殊符号后再加上一个 "?" 号）

正则表达式语法（5）

- **字符边界**

- （本组标记匹配的不是字符而是位置，符合某种条件的位置）

| | |
|----|-------------|
| ^ | 与字符串开始的地方匹配 |
| \$ | 与字符串结束的地方匹配 |
| \b | 匹配一个单词边界 |

- \b匹配这样一个位置：前面的字符和后面的字符不全是\w

正则表达式的匹配模式

- **IGNORECASE 忽略大小写模式**

- 匹配时忽略大小写。
- 默认情况下，正则表达式是要区分大小写的。

- **SINGLELINE 单行模式**

- 整个文本看作一个字符串，只有一个开头，一个结尾。
- 使小数点 "." 可以匹配包含换行符 (\n) 在内的任意字符。

- **MULTILINE 多行模式**

- 每行都是一个字符串，都有开头和结尾。
- 在指定了 MULTILINE 之后，如果需要仅匹配字符串开始和结束位置，可以使用 \A 和 \Z

正则表达式语法（6）·

• 选择符和分组

| 表达式 | 作用 |
|------------------------|---|
| 分支结构 | 左右两边表达式之间 "或" 关系，匹配左边或者右边 |
| () 捕获组 | (1). 在被修饰匹配次数的时候，括号中的表达式可以作为整体被修饰 (2). 取匹配结果的时候，括号中的表达式匹配到的内容可以被单独得到 (3). 每一对括号会分配一个编号，使用 () 的捕获根据左括号的顺序从 1 开始自动编号。捕获元素编号为零的第一个捕获是由整个正则表达式模式匹配的文本 |
| (?:Expression) 非捕获组 | 一些表达式中，不得不使用()，但又不需要保存()中子表达式匹配的内容，这时可以用非捕获组来抵消使用()带来的副作用。 |

• 反向引用（\nnn）

- 每一对()会分配一个编号，使用 () 的捕获**根据左括号的顺序从 1 开始自动编号**。
- 通过反向引用，可以对**分组已捕获的字符串**进行引用。

正则表达式语法（7）·

• 预搜索(零宽断言)

- 只进行子表达式的匹配，匹配内容不计入最终的匹配结果，是零宽度
- 这个位置应该符合某个条件。判断当前位置的前后字符，是否符合指定的条件，但不匹配前后的字符。**是对位置的匹配。**
- 正则表达式匹配过程中，如果子表达式匹配到的是字符内容，而非位置，并被保存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。占有字符还是零宽度，是针对匹配的内容是否保存到最终的匹配结果中而言的。

| | |
|----------|-----------------------|
| (?=exp) | 断言自身出现的位置的后面能匹配表达式exp |
| (?<=exp) | 断言自身出现的位置的前面能匹配表达式exp |
| (?!exp) | 断言此位置的后面不能匹配表达式exp |
| (?<!exp) | 断言此位置的前面不能匹配表达式exp |

课堂练习1!

• 电话号码验证

- (1)电话号码由数字和"-"构成
- (2)电话号码为7到8位
- (3)如果电话号码中包含有区号，那么区号为三位或四位, 首位是0.
- (4)区号用"-"和其他部分隔开
- (5)移动电话号码为11位
- (6)11位移动电话号码的第一位和第二位为"13 " , " 15" , " 18"

课堂练习2!

- **电子邮件地址验证**

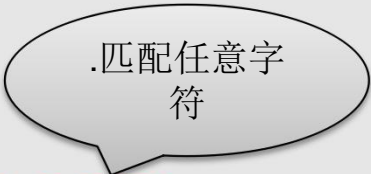
- 1.用户名：字母、数字、中划线、下划线组成。
- 2.@
- 3.网址：字母、数字组成。
- 4. 小数点：.
- 5. 组织域名：2-4位字母组成。
- 不区分大小写

常用正则表达式列表

| | |
|-----------|---|
| 匹配中文字符 | <code>[\u4e00-\u9fa5]</code> |
| 匹配空白行 | <code>\n\s*\r</code> |
| 匹配HTML标记 | <code><(\S*?)[^>]*>.*?</\1> <.*? /></code> |
| 匹配首尾空白字符 | <code>^\s* \s*\$</code> |
| 匹配Email地址 | <code>\w+([-+.] \w+)*@ \w+([-.] \w+)*\ . \w+([-.] \w+)*</code> |
| 匹配网址URL | <code>[a-zA-z]+://[^\s]*</code> |
| 匹配国内电话号码 | <code>\d{3}-\d{8} \d{4}-\d{7}</code> |
| 匹配腾讯QQ号 | <code>[1-9][0-9]{4,}</code> |
| 匹配中国邮政编码 | <code>[1-9]\d{5}(?! \d)</code> |
| 匹配身份证 | <code>\d{15} \d{18}</code> |
| 匹配ip地址 | <code>\d+\ . \d+\ . \d+\ . \d+</code> |

其他妙用

- 开发环境和文本编辑器中使用正则
 - eclipse
 - Notepad++
 - Editplus
 - UltraEdit
- 数据库中也可以使用正则
 - Mysql5.5以上
 - Oracle10g以上
 - 例如：
 - ```
SELECT prod_name
FROM products
WHERE prod_name REGEXP '.000'
```



.匹配任意字符

# JAVA程序中使用正则表达式

- 相关类位于：**java.util.regex**包下面
- **类 Pattern**：
  - 正则表达式的编译表示形式。
  - `Pattern p = Pattern.compile(r,int);` //建立正则表达式，并启用相应模式
- **类 Matcher**：
  - 通过解释 Pattern 对 character sequence 执行匹配操作的引擎
  - `Matcher m = p.matcher(str);` //匹配str字符串