# 正则表达式

## 一、在线图形化工具

<https://regexper.com/>

## 二、表达式

|  |  |  |
| --- | --- | --- |
| .（点） |  | 任意字符 |
| \d |  | 数字 |
| \D |  | 非数字 |
| \d? |  | 零次或一次 |
| \d+ |  | 循环一次或多次 |
| \d{n}  \d{3,5} |  | 循环N次  循环3~5次 |
| \d\* |  | 任意次 |
| \b |  | 单词边界 |
| \B |  | 非单词边界 |
| ^ |  | 开始 |
| $ |  | 结束 |
| \s |  | 匹配任何空白符包括空格制表符等 |
| \S |  | 非空白字符 |
| \w |  | 包括下划线的任何单词字符 |
| [a-z] |  | 查找方括号里面的字符 |
| [^a-z] |  | 查找不在方括号之间的字符 |
| \W |  | 非单词字符 |
| \n |  | 换行符 |
| \r |  | 回车符 |
| \t |  | 制表符 |
| ｜ |  | 指明两项之间的一个选择。要匹配 |，请使用 \|。 |

\*、+与？限定符都是贪婪的，它们会尽可能多的匹配字符，只要在它们的后面加上一个？就可以实现非贪婪的（懒惰）或最小匹配

.\*表示尽可能多的匹配字符

.\*?表示尽可能少的匹配

如：字符串aabab,用贪婪模式匹配a.\*b得到aabab,用懒惰模式匹配a.\*?b得到aab

如：贪婪型匹配的一个例子：

文本：This offer is not available to customers living in **<B>AK</B> and <B>HI</B>**

正则表达式: <B>.\*</B>

其匹配的结果是<B>AK</B> and <B>HI</B>

正则表达式修改为<B>.\*?</B>

其匹配的结果:<B>AK</B> <B>HI</B>

|  |  |  |  |
| --- | --- | --- | --- |
| **.** | **[ ]** | **^** | **$** |

四个字符是所有语言都支持的正则表达式

　　**等价：**

等价是等同于的意思，表示同样的功能，用不同符号来书写。

?,\*,+,\d,\w 都是等价字符  
　　 ?等价于匹配长度{0,1}  
　　 \*等价于匹配长度{0,}   
　　 +等价于匹配长度{1,}  
　 　\d等价于[0-9]

\D等价于[^0-9]  
　 　\w等价于[A-Za-z\_0-9]

\W等价于[^A-Za-z\_0-9]。

**常用运算符与表达式：**^ 开始  
　　（） 域段  
　 　[] 包含,默认是一个字符长度  
　 　[^] 不包含,默认是一个字符长度  
　 　{n,m} 匹配长度   
　 　. 任何单个字符(\. 字符点)  
　　 | 或  
　 　\ 转义  
　 　$ 结尾  
　 　[A-Z] 26个大写字母  
　 　[a-z] 26个小写字母  
　 　[0-9] 0至9数字

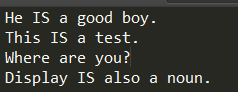
[A-Za-z0-9] 26个大写字母、26个小写字母和0至9数字  
　　， 分割  
　　.  
　　  
　　**分割语法：**  
　　[A,H,T,W] 包含A或H或T或W字母  
　　[a,h,t,w] 包含a或h或t或w字母  
　　[0,3,6,8] 包含0或3或6或8数字

　　**语法与释义：**基础语法 "^([]{})([]{})([]{})$"  
　　正则字符串 = "开始（[包含内容]{长度}）（[包含内容]{长度}）（[包含内容]{长度}）结束"    
　　?,\*,+,\d,\w 这些都是简写的,完全可以用[]和{}代替，在(?:)(?=)(?!)(?<=)(?<!)(?i)(\*?)(+?)这种特殊组合情况下除外。

## 三、案例

**1.单词边界\b**

需求把IS替换成is

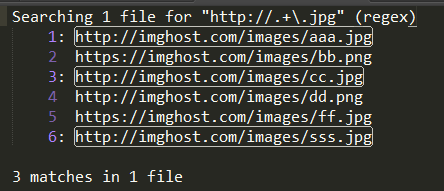
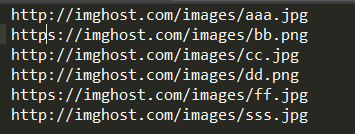


\bis\b

图解：



**2.查找http开头jpg结尾的**

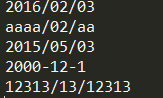
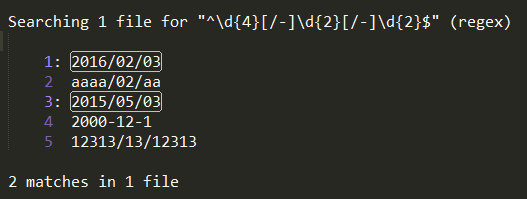


表达式：http:\/\/.+\.jpg

图解

“.”表示任意字符 “+”表示出現一次或多次

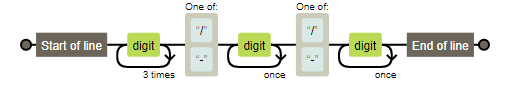
**3.查找日期**

表达式：^\d{4}[/-]\d{2}[/-]\d{2}$

^表示开头 $表示结尾 []表示或

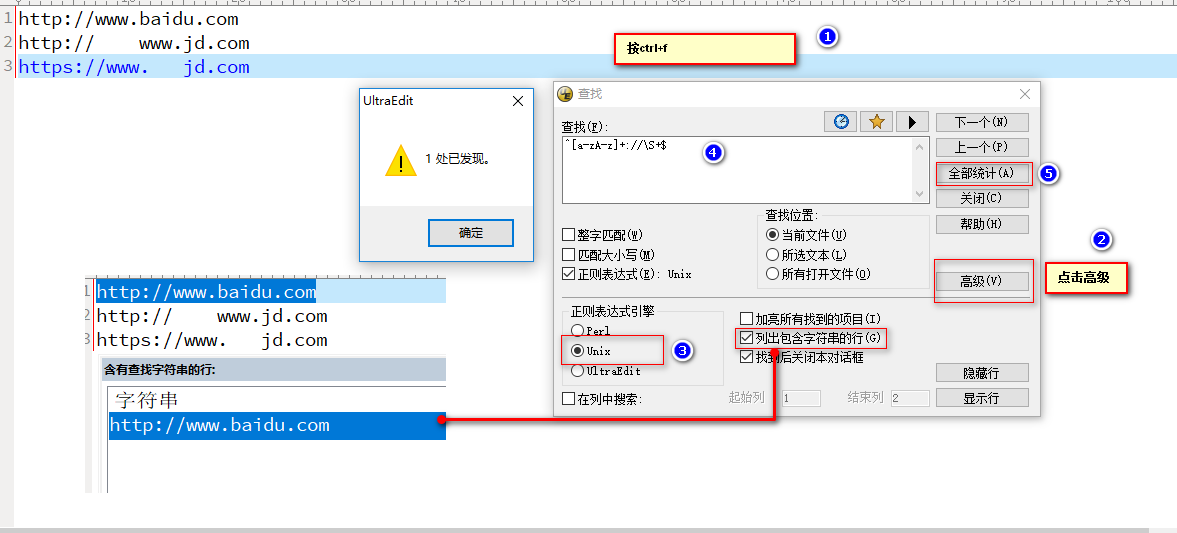
图解：



**4.匹配网址**

表达式：^[a-zA-z]+://\S+$

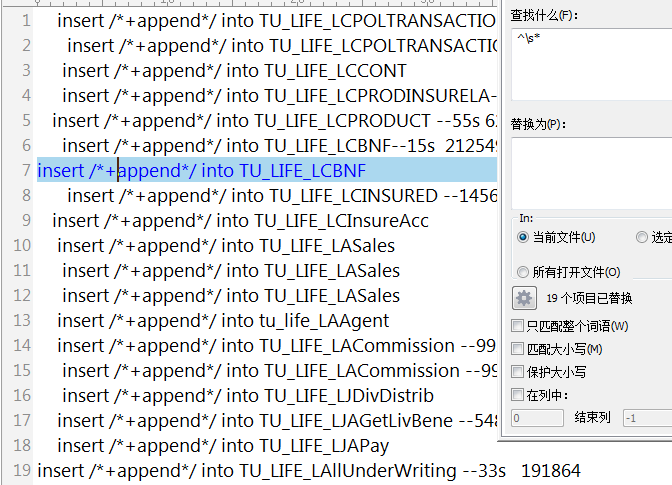
‘\S’任何非空白字符



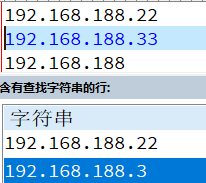
**5.去除行首的空格 ^\s\***

\s\*$ 行尾

^\s\*|\s\*$ 行首或行尾



**6.匹配IP地址： \d+\.\d+\.\d+\.\d+**



**7.匹配帐号是否合法(字母开头，允许5-16字节，允许字母数字下划线)**

^[a-zA-Z][a-zA-Z0-9\_]{4,15}$

**8.匹配腾讯QQ号**

^[1-9]\d{4,10}$

**9匹配用户名**

以字母开头+数字/字母/下划线

^[a-zA-Z][a-zA-z1-9\_]+$

**10手机号码**

^1[3|4|5|8][0-9]\d{8}$

**11.18位身份证号**

18位：

^[1-9]\d{5}(18|19|([23]\d))\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{3}[0-9X

15位：

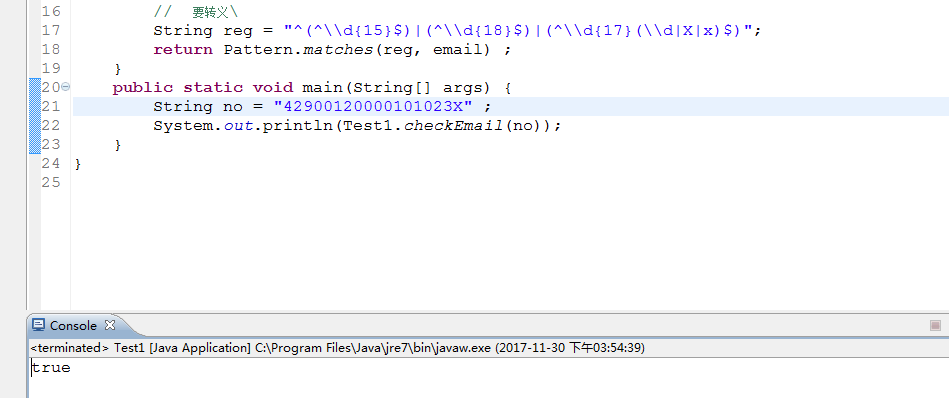
^[1-9]\d{5}\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{2}$

总：

^[1-9]\d{5}(18|19|([23]\d))\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{3}[0-9Xx]$)|(^[1-9]\d{5}\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{2}$

简单的例子

java.util.regex.Pattern.matches();



**12.只能输入汉字：**

^[\u4e00-\u9fa5]{0,}$

**13. 匹配空白行的正则表达式：**

\n\s\*\r

**14. 匹配HTML标记的正则表达式：**

<(\S\*?)[^>]\*>.\*?</>|<.\*? />

### 1.在java中使用正则

##### 取掉字符串中的空格 ：

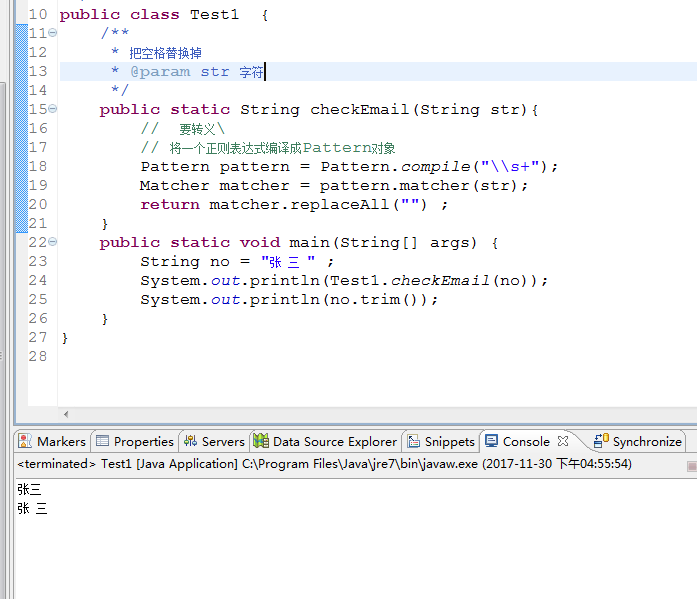
代码：

// 将一个正则表达式编译成Pattern对象

Pattern pattern = Pattern.*compile*("\\s+");

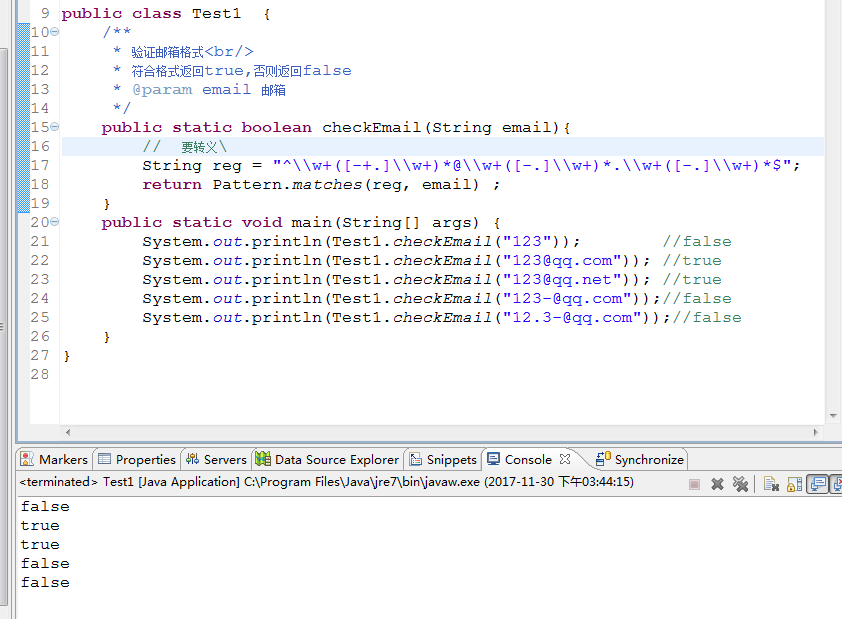
Matcher matcher = pattern.matcher(str);

**return** matcher.replaceAll("") ;



##### 验证邮箱:

^\w+([-+.]\w+)\*@\w+([-.]\w+)\*.\w+([-.]\w+)\*$



### 2.在javascript中使用正则表达式

#### Search()

<script type="text/javascript">

function check (){

var str = document.getElementById('str').value;

//search() 方法 用于检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串，并返回子串的起始位置。

//下标从0开始

if(str.search(/敏感字符/)>=0){

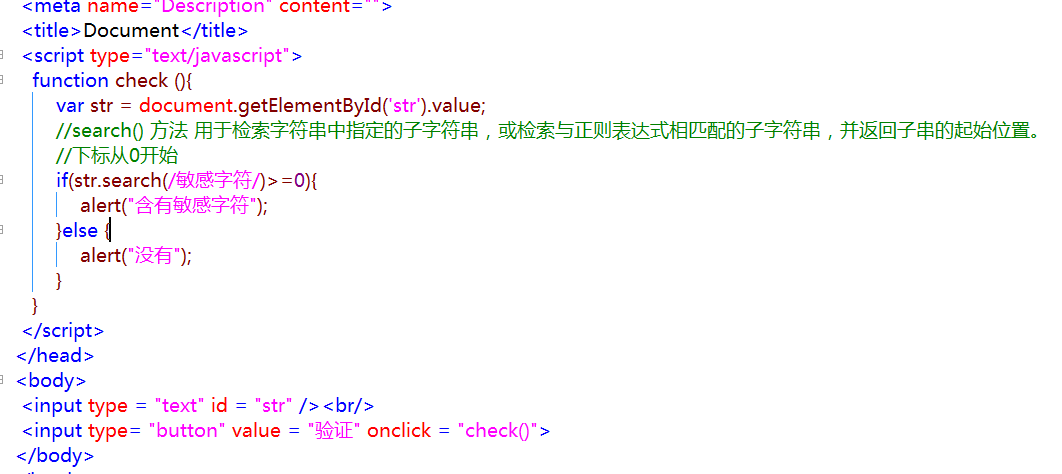
alert("含有敏感字符");

}else {

alert("没有");

}

}



结果：





匹配是否含有#- or 空格 这类符号 i 忽略大小写

/i (忽略大小写)  
 /g (全文查找出现的所有匹配字符)  
 /gi(全文查找、忽略大小写)  
 /ig(全文查找、忽略大小写)

////var reg = /^[^#|^-|^or|\s]\*$/**i**; --这样的话，会导致只要含有or(aaaoroo)都是false

var s = str.search(/#|\bor\b|-|'|"|;/);

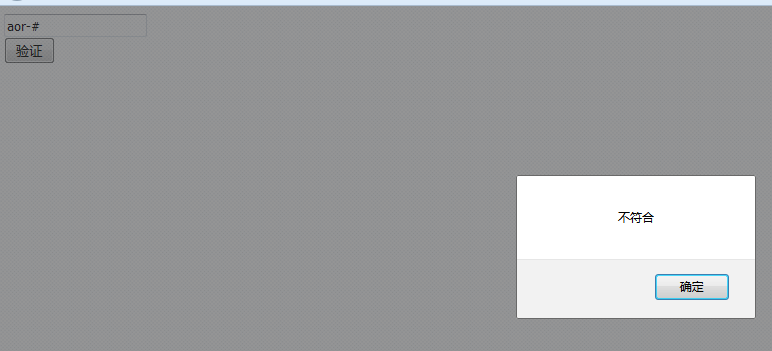
if(s>=0){

alert("不符合");

}else {

alert("符合");

}





#### Replace();



/\*是否带有小数\*/

function isDecimal(strValue ) {

var objRegExp= /^\d+\.\d+$/;

return objRegExp.test(strValue);

}

/\*校验是否中文名称组成 \*/

function ischina(str) {

    var reg=/^[\u4E00-\u9FA5]{2,4}$/; /\*定义验证表达式\*/

    return reg.test(str); /\*进行验证\*/

}

/\*校验是否全由8位数字组成 \*/

function isStudentNo(str) {

    var reg=/^[0-9]{8}$/; /\*定义验证表达式\*/

    return reg.test(str); /\*进行验证\*/

}

/\*校验电话码格式 \*/

function isTelCode(str) {

    var reg= /^((0\d{2,3}-\d{7,8})|(1[3584]\d{9}))$/;

    return reg.test(str);

}

/\*校验邮件地址是否合法 \*/

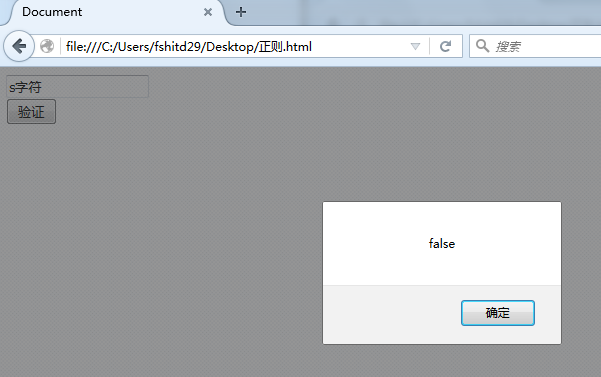
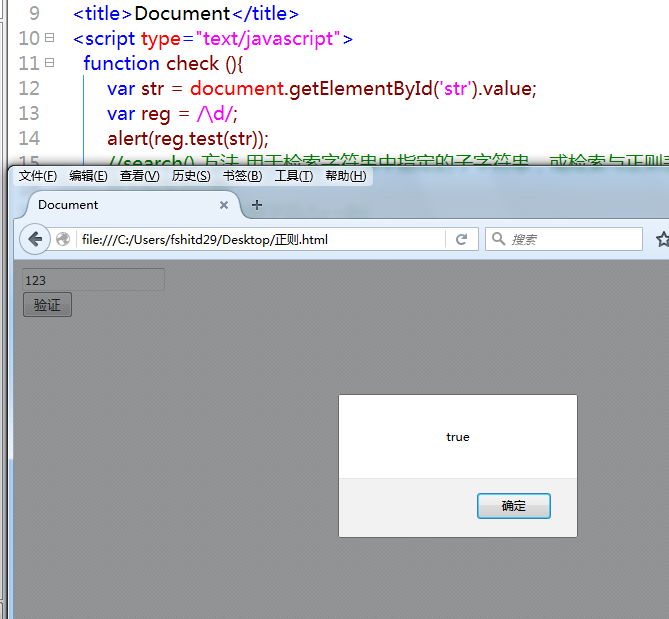
function IsEmail(str) {

    var reg=/^([a-zA-Z0-9\_-])+@([a-zA-Z0-9\_-])+(\.[a-zA-Z0-9\_-])+/;

    return reg.test(str);

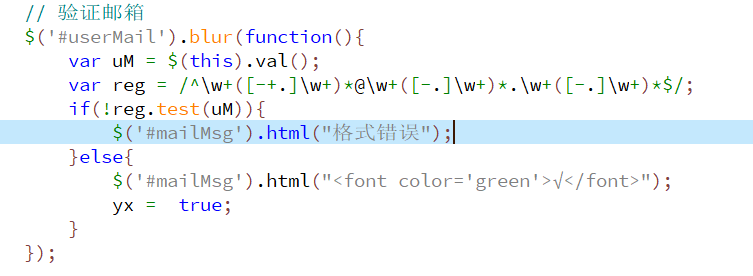
}

#### test() 方法用于检测一个字符串是否匹配某个模式，如果字符串中含有匹配的文本，则返回 true，否则返回 false。



\w 任何单词字体

/^\w+([-+.]\w+)\*@\w+([-.]\w+)\*.\w+([-.]\w+)\*$/



### 3．在oracle中使用正则表达式

在oracle中使用正则表达式的函数：

REGEXP\_SUBSTR(srcstr, pattern [, position [, occurrence [, match\_option]]])

注：srcstr ：要匹配的字符串

Pattern：正则表达式

position ：从第几个字符开始正则表达式匹配。（默认为1）

occurrence ：标识第几个匹配组。（默认为1）

match\_option ：取值范围

i：大小写不敏感；

c：大小写敏感；

n：点号 . 不匹配换行符号；

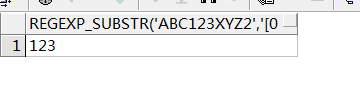
m：多行模式；

x：扩展模式，忽略正则表达式中的空白字符。

例子：

1.匹配” ABC123XYZ2”这个字符串中的数字

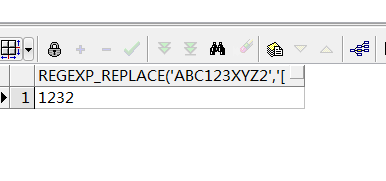
select regexp\_substr('ABC123XYZ2','[0-9]+') from dual;



如果要匹配字符串中所有数字:

REGEXP\_REPLACE(source\_char, pattern [, replace\_string [, position [, occurrence [, match\_parameter ] ] ] ] )

select regexp\_replace('ABC123XYZ2','[^0-9]','') from dual;



替换电话号码不规则的字段

00432-1231一地在23123\*\*\* ：把- 汉字 \* 替换掉

像\* / \ -这类的关键字段要记得转义

select REGEXP\_REPLACE('00432-1231一地在23123\*\*\*', '\\*|\-|\－|[^0-9]', '')

from dual

