1. 常用正则表达式
2. 【数量：单个】字符匹配

·任意字符(a)：表示由任意字符(a)组成

·\\：匹配“\”;

·\n：匹配换行；

·\t：匹配指标符号;

1. 【数量：单个】字符集（可以从中任选一个字符）

·[abc]：表示可能时字符abc中的任意一个；

·[^abc]：表示不是abc中的任意一个；

·[a-zA-Z]：表示由任意一个字符组成，不区分大小写；

·[0-9]：表示由一位数字组成；

1. 【数量：单个】简化的字符集

· .:点表示任意的一个字符

·\d：等价于[0-9]范围(正则时我们通常写成 [\\d](\\\\d) ,因为\\代表的是\)

·\D：等价于[^0-9]范围

·\s：匹配任意的空格，可能是空格，制表符，换行；

·\S：匹配任意的非空格数据；

·\w：匹配任意的数字，字母，下划线，等价于“[a-zA-Z\_0-9]”

·\W：匹配非\w

1. 边界匹配:

·^:匹配边界开始；

·$：匹配边界结束；

1. 数量表示，默认情况下只有添加了数量单位才可以匹配多位字符:

·表达式？：该正则可以出现0次或者1次

·表达式\*：该正则可以出现0次或多次

·表达式+：该正则可以出现1次或多次

·表达式{n}：表达式的长度正好为n

·表达式{n,}：表达式的长度为n以上

·表达式{n,m}：表达式的长度为n~m

范例:匹配多个字符

|  |
| --- |
|  |

1. String类对正则表达式的支持

在进行正则表达式大部分处理的情况都是基于String类完成的，并且String类提供由如下方法支持:

|  |
| --- |
|  |

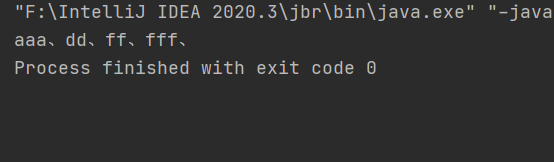
下面通过一些具体的范例来对正则的使用进行说明。

范例:删除所有非字母数字

|  |
| --- |
| package com.company;  public class JavaDemo {  public static void main(String[] args) {  //要判断的数据，删除所有非字母，数字  String str = "asdhaf12345#@131sdf#$%#$%3" ;  String regex = "\\W+";  System.out.println(str.replaceAll(regex,""));  }  } |

范例:实现字符串的拆分

|  |
| --- |
| package com.company;  public class JavaDemo {  public static void main(String[] args) {  //要判断的数据，删除所有非字母，数字  String str = "aaa111dd44ff556fff44" ;  String regex = "\\d+";  String[] results = str.split(regex);  for (int i = 0; i < results.length; i++) {  System.out.print(results[i]+"、");  }  }  } |



1. java.util.regex开发包

虽然在大部分的情况下都可以利用String类实现正则的操作，但是也有一些情况需要用到java.util.regex开发包中提供的正则处理类，在这个包中一共定义两个类，一个Pattern类、Matcher类

1. Pattern类提供有正则表达式的编译处理:public static Pattern compile(String regex) ;
2. 同时也提供有字符串的拆分操作: public String[] split(CharSequence input );

范例：字符串拆分

|  |
| --- |
| package com.company;  import java.util.regex.Pattern;  public class JavaDemo {  public static void main(String[] args) {  //要判断的数据，删除所有非字母，数字  String str = "jsfajs1#@$#%shdf@#$sdfsk$%%" ;  String regex = "[^a-zA-Z0-9]+";  Pattern pattern = Pattern.compile(regex);  String[] results = pattern.split(str);  for (int i = 0; i < results.length; i++) {  System.out.print(results[i]+"、");  }  }  } |

1. Matcher类，实现了正则匹配的处理类，这个类的实例化对象依靠Pattern类完成:
   1. Pattern类提供的方法:public Match matcher(CharSequence input);

当获取Matcherl类的对象之后就可以利用该类的方法进行操作

* 1. 正则匹配: public boolean matchers();
  2. 字符串的替换:public String replaceAll(String replacem)

1. 如果是纯粹的拆分，替换，匹配三种操作根本用不到java.util.regex 开发包，只依靠String就可以实现了。但是Matcher类里面提供有一种分组的功能，而这种功能是String类不具备的。

范例:

|  |
| --- |
| package com.company;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class JavaDemo {  public static void main(String[] args) {  //取出#{}中的所有内容  String str = "INSERT INTO dept(deptno ,dname,loc) VALUES #{deptno},#{dname},#{loc}" ;  String regex = "#\\{\\w+}";  Pattern pattern = Pattern.compile(regex);  Matcher matcher = pattern.matcher(str);  while(matcher.find()){  System.out.println(matcher.group(0).replaceAll("[#\\{\\}]",""));  }  }  } |

Java.util.regex开发包，如果不是进行一些更为复杂的正则处理是很难使用到的，而String类所提供的功能只适合正则的基本操作，而其中正则的验证是这些基础操作中最重要的部分。