数据工具类，用于处理string，number，list等数据。Maven添加

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.9</version>

</dependency>

# ArrayUtils数组工具

支持int、Integer、byte、Byte、double、Double、float、Float、long、Long、short、Short、char、Char构建数组

byte[] bytes = ArrayUtils.EMPTY\_BYTE\_ARRAY;

Byte[] Bytes = ArrayUtils.EMPTY\_BYTE\_OBJECT\_ARRAY;

int[] ints = ArrayUtils.EMPTY\_INT\_ARRAY;

Integer[] Ints = ArrayUtils.EMPTY\_INTEGER\_OBJECT\_ARRAY;

long[] longs = ArrayUtils.EMPTY\_LONG\_ARRAY;

Long[] Longs = ArrayUtils.EMPTY\_LONG\_OBJECT\_ARRAY;

double[] doubles = ArrayUtils.EMPTY\_DOUBLE\_ARRAY;

Double[] Doubles = ArrayUtils.EMPTY\_DOUBLE\_OBJECT\_ARRAY;

short[] shorts = ArrayUtils.EMPTY\_SHORT\_ARRAY;

Short[] Shorts = ArrayUtils.EMPTY\_SHORT\_OBJECT\_ARRAY;

char[] chars = ArrayUtils.EMPTY\_CHAR\_ARRAY;

Character[] characters = ArrayUtils.EMPTY\_CHARACTER\_OBJECT\_ARRAY;

String[] strings = ArrayUtils.EMPTY\_STRING\_ARRAY;

ArrayUtils.add(arr,index,value);

向数组的末尾添加1个数

int[] arr = ArrayUtils.add(new int[]{1,2,3},4);//1,1,2,3,4

ArrayUtils.addAll(arr,...nums);

向数组的末尾添加n个数

int[] arr = ArrayUtils.add(new int[]{1,2,3},4,5,6,7,8);//1,1,2,3,4,5,6,7,8

ArrayUtils.add(arr,index,value);

向数组的对应位置插入1个数

int[] arr = ArrayUtils.add(new int[]{0,1,2,3,4},1,41);//0,41,1,2,3,4

ArrayUtils.clone(arr);

深拷贝（单独开辟内存）

int[] arr = ArrayUtils.clone(new int[]{1,2,3,4,5});

ArrayUtils.insert(index,arr,value);

向数组的对应位置插入1个数

ArrayUtils.indexOf(arr,num)、ArrayUtils.indexOf(arr,num,startIndex)

查询某一数在数组中位置

Integer[] array = {1,2,3,4,5,6};

int index = ArrayUtils.indexOf(array, 2);

ArrayUtils.lastIndexOf(arr, num);

num在arr中最后一次出现的位置

Integer[] array = {1,2,3,4,5,6,2};

int index = ArrayUtils.lastIndexOf(array, 2);

ArrayUtils.isEmpty(arr)

验证arr是否为[],是为true

ArrayUtils.isNotEmpty(arr)

验证arr是否为[]，否为true

ArrayUtils.isArrayIndexValid(array, index);

验证index是否在索引array的安全范围。

Boolean index = ArrayUtils.isArrayIndexValid(array, 19);

ArrayUtils.swap(array, value1, value2);

交换数组中2个位置的值

int[] arr = ArrayUtils.swap(new int[]{1,2,3},0,1);//2,1,3

ArrayUtils.contains(array, num);

检验array中是否存在num

int[] array = new int[]{1,2,3}

Boolean b = ArrayUtils.contains(array, 10);//false

ArrayUtils.remove(arr,index)

删除arr中下标index的数据

ArrayUtils.removeAll(array, ...nums);

删除arr中下标处于...nums中的数据

int[] array = {1,2,3,4,5};

array = ArrayUtils.removeAll(array, 1,2,3);//1 5

ArrayUtils.reverse(arr)

颠倒arr(改变原数组)

ArrayUtils.reverse(new int[]{1,2,3,4,5});//5 4 3 2 1

# StringUtils字符串工具

StringUtils.abbreviate(str, maxLength);

缩略str内容,(maxLength内容必须大于3)

String res = StringUtils.abbreviate( "i am ok, how about you?", 10);//i am ok...

StringUtils.abbreviate(str, offset,maxLength);

缩略str内容,(maxLength内容必须大于3；offset必须大于5，从第六个开始算第一个)

String res = StringUtils.abbreviate( "i am ok, how about you?",7,10);//...ok, ...

StringUtils.abbreviate(str, abbrevMarker,maxLength);

缩略str内容,(maxLength内容必须大于3；用abbrevMarker替代...)

String res = StringUtils.abbreviate( "i am ok, how about you?","--",10);//i am ok,--

StringUtils.capitalize(str)

str首字母大写

String res = StringUtils.capitalize("abcdefg");//Abcdefg

StringUtils.countMatches(str,str1)

str1在str中出现的次数

int res = StringUtils.countMatches("this is a good car","i");//2

StringUtils.contains(str,int ASCⅡ);

验证str中是否包含用ASCⅡ表示的字符，返回boolean

Boolean res = StringUtils.contains("123", 65);

StringUtils.contains(str, str1);

验证str中是否包含str1，返回boolean

Boolean res = StringUtils.contains("123", "1");

StringUtils.containsAny(str, ...strs);

验证str中是否包含strs中的字符串，返回boolean

Boolean res = StringUtils.containsAny("1234567", "1","2","3");

StringUtils.containsAny(str, ...chars);

验证str中是否包含chars中字符，返回boolean

Boolean res = StringUtils.containsAny("1234567", '1','2','3');

StringUtils.containsNone(str1, str1);

验证str中是否包含str1，包含返回false，不包含返回true，返回boolean

Boolean res = StringUtils.containsNone("abc", "d");

StringUtils.containsNone(str, ...chars)

验证str中是否包含chars中，包含返回false，不包含返回true，返回boolean

Boolean res = StringUtils.containsNone("abc", 'k','y','d');

StringUtils.containsIgnoreCase(str, str1);

验证str中是否包含str1（忽略大小写），返回boolean

Boolean res = StringUtils.containsIgnoreCase("abc", "A");

StringUtils.deleteWhitespace(str)

不改变str，删除str中空格，返回新字符串

String res = StringUtils.deleteWhitespace("a b c");//abc

StringUtils.startsWith(str, str1);

验证str是否以str1开始，返回boolean

Boolean res = StringUtils.startsWith("abcd", "a");

StringUtils.endsWith(str, str1);

验证str是否以str1结束，返回boolean

Boolean res = StringUtils.endsWith("abcd", "a");

StringUtils.startsWithAny(str,...strs);

验证str是否以str\_1~str\_n中某一个开头，返回boolean

Boolean res = StringUtils.startsWithAny("str", "str\_1","str\_2",...,"strn");

StringUtils.endsWithAny(str,...strs);

验证str是否以str\_1~str\_n中某一个结束，返回boolean

Boolean res = StringUtils.endsWithAny("str", "str\_1","str\_2",...,"strn");

StringUtils.startsWithIgnoreCase(str,str1);

验证str是否以str1开始（忽略大小写），返回boolean

Boolean res = StringUtils.startsWithIgnoreCase("abcd", "A");

StringUtils.endsWithIgnoreCase(str,str1);

验证str是否以str1结束（忽略大小写），返回boolean

Boolean res = StringUtils.endsWithIgnoreCase("abcd", "A");

StringUtils.containsOnly(str,str1)

验证str1中是否含有str，返回boolean

Boolean res = StringUtils.containsOnly("abc", "abc,def,ghi,jkl");

StringUtils.containsOnly(str,char)

验证char中是否含有str，返回boolean

Boolean res = StringUtils.containsOnly("abc", 'a');

StringUtils.containsWhitespace(str);

验证str中是否含空格，有则返回true，不含返回false。

Boolean res = StringUtils.containsWhitespace("abcafljals ");

StringUtils.defaultIfBlank(str,str1)

str若为null或" "，返回str1。

String res = StringUtils.defaultIfBlank(null, "DEFO");

StringUtils.defaultIfEmpty(str,str1)

str若为""，返回str1

String res = StringUtils.defaultIfBlank("", "DEFO");

StringUtils.defaultString(str)

str若为null，返回""

String res = StringUtils.defaultString(null);

StringUtils.defaultString(str,str1)

str若为null，返回str1

String res = StringUtils.defaultString(null,"hello");

StringUtils.equals(str, str1);

验证str与str1是否相等，返回boolean

Boolean res = StringUtils.equals("a", "b");

StringUtils.equalsIgnoreCase(str, str1);

验证str与str1是否相等（忽略大小写），返回boolean

Boolean res = StringUtils.equalsIgnoreCase("a", "A");

StringUtils.equalsAny(str, ...str1)

验证str1中是否存在与str相等的项，返回boolean

Boolean res = StringUtils.equalsAny("a", "a","c");

StringUtils.equalsAnyIgnoreCase(str, ...str1)

验证str1中是否存在与str相等的项（忽略大小写），返回boolean

StringUtils.equalsAnyIgnoreCase("a", "a","c")

StringUtils.getDigits(str)

提取str中的数字并返回

String res = StringUtils.getDigits("af123defg456");//res:123456

StringUtils.getCommonPrefix(...str1);

提取str1中共同的前缀并返回

String res = StringUtils.getCommonPrefix("abc","abd","abe","abf");//返回ab

StringUtils.indexOf(str,str1)

str1在str中开始出现的位置（从0开始，若str1在str中不存在则返回-1）返回int

int i = StringUtils.indexOf("abc", "c");

StringUtils.indexOf(str,int ASCⅡ)

以ASCⅡ表示的字符在str中的位置（从0开始，若不存在则返回-1）返回int

int i = StringUtils.indexOf("abcdefg", 97);

StringUtils.indexOf(str, str1, int startIndex);

从str[startIndex]位置开始查找str1第一次出现的位置，返回int（参考点还是str的第一个开头）

int i = StringUtils.indexOf("abcdefg", "c", 4);

StringUtils.indexOf(str, int ASCⅡ, int startIndex);

从str[startIndex]位置开始查找以ASCⅡ表示的字符第一次出现的位置，返回int

int i = StringUtils.indexOf("abcdefg", 99, 4);

StringUtils.indexOfIgnoreCase(str, str1);

str1在str中开始出现的位置（忽略大小写，从0开始，若str1在str中不存在则返回-1）返回int

StringUtils.indexOfIgnoreCase("abcdefg", "C");

StringUtils.indexOfIgnoreCase(str, str1, int startIndex);

从str[startIndex]位置开始查找str1第一次出现的位置（忽略大小写），返回int

StringUtils.indexOfIgnoreCase("abcdefg", "C");

StringUtils.indexOfAny(str, ...str1);

查找str1中某一个字符串在str第一次出现的位置，返回int

int i = StringUtils.indexOfAny("abcdefg","k","b");

StringUtils.indexOfAny(str, ...char);

查找char中某一个字符在str第一次出现的位置，返回int

int i = StringUtils.indexOfAny("abcdefg",'k','b');

StringUtils.indexOfAnyBut(str,...char)

查找字符串str中不在字符数组char中的第一个字符的位置，返回int

int res = StringUtils.indexOfAnyBut("abcd", 'a','b','c');//d不存在于['a','b','c']中

StringUtils.indexOfAnyBut(str, str1);//cd不存在于ab中

查找str中不在str1中的第一个字符的位置，返回int

int res = StringUtils.indexOfAnyBut("abcd", "ab");//cd不存在于ab中

StringUtils.indexOfDifference(str,str1);

比对str和str1中开始不同的起始位置，返回int

int res = StringUtils.indexOfDifference("acdefg","acdehi");//4

StringUtils.indexOfDifference(...str);

比对str中开始不同的起始位置，返回int

int res = StringUtils.indexOfDifference("acdefg","acdehi","acdehk");//4

StringUtils.join(str[],str1)

将str数组用str1连接返回一个新字符串。

String res = StringUtils.join(new String[]{"aaa","bbb","ccc"},"-");//aaa-bbb-ccc

StringUtils.join(str[],char)

将str数组用str1连接返回一个新字符串。

String res = StringUtils.join(new String[]{"aaa","bbb","ccc"},'-');//aaa-bbb-ccc

StringUtils.join(List<String> list,char)

将序列list用str1连接返回一个新字符串。

String res = StringUtils.join(arrayList,'-');//aaa-bbb-ccc

StringUtils.join(List<Integer> list,char)

将序列list用str1连接返回一个新字符串。

String res = StringUtils.join(num\_arr,'-');//0-1-2

StringUtils.uncapitalize(str)

首字母小写

String res = StringUtils.uncapitalize("Java");//java

# NumberUtils数字工具

NumberUtils.compare(num1,num2)

比较num1是否大于num2，>返回1，<返回-1，=返回0。num可以是byte，int，long，short类型。

int integer = NumberUtils.compare(-1,2);

String转num

|  |  |
| --- | --- |
| NumberUtils.createInteger(str) | String转int |
| NumberUtils.createFloat(str) | String转float |
| NumberUtils.createDouble(str) | String转double |
| NumberUtils.createBigDecimal(str) | String转bigdecimal |
| NumberUtils.createBigInteger(str) | String转biginteger |
| NumberUtils.createLong(str) | String转long |
| NumberUtils.createNumber(str) |  |

|  |  |  |
| --- | --- | --- |
|  |  | Obj可选 |
| NumberUtils.toInt(obj) | obj转int | string |
| NumberUtils.toByte(obj) | obj转float | string |
| NumberUtils.toShort(obj) | obj转double | BigDecimal，String |
| NumberUtils.toDouble(obj) | obj转bigdecimal | BigDecimal，String |
| NumberUtils.toFloat(obj) | obj转biginteger | String |
| NumberUtils.toLong(obj) | obj转long | String |
| NumberUtils.toScaledBigDecimal(obj) |  | BigDecimal，Double，float，String |

抽取极值

|  |  |  |
| --- | --- | --- |
|  |  | Obj可选 |
| NumberUtils.max(...num) | 返回最大值 | 所有数字的数据类型 |
| NumberUtils.min(...num) | 返回最小值 | 所有数字的数据类型 |

# RandomUtils随机数生成工具

|  |  |
| --- | --- |
|  | 解释 |
| RandomUtils.nextInt() | 随机返回int型值 |
| RandomUtils.nextInt(start,end) | 随机返回[start,end]范围内的int型值 |
| RandomUtils.nextBoolean() | 随机返回boolean型值 |
| RandomUtils.nextByte(int count) | 随机返回长度为count的byte数组 |
| RandomUtils.nextDouble() | 随机返回double型值 |
| RandomUtils.nextDouble(start,end) | 随机返回[start,end]范围内的布尔型值 |
| RandomUtils.nextFloat() | 随机返回Float型值 |
| RandomUtils.nextFloat(start,end) | 随机返回[start,end]范围内的float型值 |
| RandomUtils.nextLong() | 随机返回long型值 |
| RandomUtils.nextLong(start,end) | 随机返回[start,end]范围内的long型值 |

# DateUtils日期格式类

|  |  |
| --- | --- |
|  | 解释 |
| DateUtils.addMilliseconds(Date,int) | 加减毫秒 |
| DateUtils.addSecond(Date,int) | 加减秒 |
| DateUtils.addMinutes(Date,int) | 加减分钟 |
| DateUtils.addHours(Date,int) | 加减小时 |
| DateUtils.addDays(Date,int) | 加减天 |
| DateUtils.addWeeks(Date,int) | 加减周 |
| DateUtils.addMonths(Date,int) | 加减月份 |
| DateUtils.addYears(Date,int) | 加减年份 |

固定值

|  |  |  |  |
| --- | --- | --- | --- |
|  | 类型 | 数值 | 解释 |
| .MILLIS\_PER\_SECOND | long | 100 | 一秒钟的毫秒数 |
| .MILLIS\_PER\_MINUTE | long | 6000 | 一分钟的毫秒数 |
| .MILLIS\_PER\_HOUR | long | 3600000 | 一小时的毫秒数 |
| .MILLIS\_PER\_DAY | long | 86400000 | 一天的毫秒数 |

Date res = DateUtils.parseDate( str , str[…] )

将str的时间转换为Date数据格式。str[…]是数据的格式范围。

Date res = DateUtils.parseDate("2020-10-10 10:10:10", new String[] {"yyyy-MM-dd","yyyy-MM-dd hh:mm:ss"});

Date res = DateUtils.ceiling(new Date(),Calendar.HOUR);

将时间向上取整，第二个参数可选Calendar. MILLISECOND, Calendar.SECOND, Calendar.MINUTE, Calendar.HOUR, Calendar.MONTH, Calendar.YEAR

Date date = DateUtils.ceiling(new Date(),Calendar.DATE);