StringUtils就这1张图，必备（二）

原创 阿_毅 于 2016-08-20 22:31:40 发布 2071 版权
分类专栏： 一步一步学开源工具包 文章标签： StringUtil

开心一笑

【男女朋友一起去逛街，
女朋友：哎哟，脚好酸哦。
男朋友很紧张：怎么了？是不是踩到柠檬了？】

视频教程

大家好，我录制的视频《Java之优雅编程之道》已经在CSDN学院发布了，有兴趣的同学可以购买观看，相信大家一定会收获到很多知识的。谢谢大家的支持……

视频地址：http://edu.csdn.net/lecturer/994

提出问题

Land3的StringUtils类如何使用？？？

解决问题

StringUtils是工作中使用最频繁的一个工具类，提供了大量丰富的字符串 操作方法，下面是所有方法的一个蓝图：

判空函数

1）StringUtils.isEmpty(String str)

2）StringUtils.isNotEmpty(String str)

3）StringUtils.isBlank(String str)

4）StringUtils.isNotBlank(String str)

5）StringUtils.isAnyBlank(CharSequence… css)

6）StringUtils.isAnyEmpty(CharSequence… css)

7）StringUtils.isNoneBlank(CharSequence… css)

8）StringUtils.isNoneEmpty(CharSequence… css)

- isNotEmpty = !isEmpty, isBlank同理；
- 容易忽略的；

```
StringUtils.isEmpty("")        = true
```

- isBlank和isEmpty区别：

```
System.out.println(StringUtils.isBlank(" "));//true

System.out.println(StringUtils.isBlank("   "));//true

System.out.println(StringUtils.isBlank("\n\t"));//true
//区别
StringUtils.isEmpty(" ")       = false
```

- isAnyBlank和isAnyEmpty是多维判空，存在一个blank或者empty既true

```
StringUtils.isAnyBlank("", "bar","foo");       = true
//注意这两个区别
StringUtils.isAnyEmpty(" ", "bar")       = false
StringUtils.isAnyEmpty(" ", "bar")       = true
```

- isNoneBlank = !isAnyBlank；isNoneEmpty同理

```
public static boolean isNoneBlank(CharSequence... css) {
    return !isAnyBlank(css);
}
```

- isWhitespace判断空白

```
StringUtils.isWhitespace(null)   = false
 StringUtils.isWhitespace("")     = true
 StringUtils.isWhitespace("  ")   = true
```

大小写函数

StringUtils.capitalize(String str)

StringUtils.uncapitalize(String str)

StringUtils.upperCase(String str)

StringUtils.upperCase(String str, locale (现场) locale)

StringUtils.lowerCase(String str)

StringUtils.lowerCase(String str, locale (现场) locale)

StringUtils.swapCase(String str)

StringUtils.isAllUpperCase(CharSequence cs)

StringUtils.isAllLowerCase(CharSequence cs)

注意点：

- capitalize首字母大写，upperCase全部转化为大写，swapCase大小写互转；

```
StringUtils.capitalize(null)  = null

 StringUtils.capitalize("")    = ""

//首字母转为大写
 StringUtils.capitalize("cat") = "Cat"
```

```
StringUtils.upperCase("aBc") = "ABC"

//大小写互转
StringUtils.swapCase("The dog has a BONE") = "tHE DOG HAS A bone"
```

- isAllUpperCase是否全部大写，isAllLowerCase是否全部小写

```
 StringUtils.isAllLowerCase("  ")   = false

 StringUtils.isAllLowerCase("abc")  = true

 StringUtils.isAllLowerCase("abC")  = false

 StringUtils.isAllLowerCase("ab c") = false

 StringUtils.isAllLowerCase("ab1c") = false

 StringUtils.isAllLowerCase("ab/c") = false
```

删除函数

StringUtils.remove(String str, char (煤焦) remove)

StringUtils.remove(String str, String remove)

StringUtils.removeEnd(String str, String remove)

StringUtils.removeEndIgnoreCase(String str, String remove)

StringUtils.removePattern(String source, String regex)

StringUtils.removeStart(String str, String remove)

StringUtils.removeStartIgnoreCase(String str, String remove)

StringUtils.deleteWhitespace(String str)

- 具体例子

```
 //删除字符
 StringUtils.remove("queued", 'u') = "qeed"

//删除字符串
StringUtils.remove("queued", "ue") = "qd"

//删除结尾匹配的字符串
StringUtils.removeEnd("www.domain.com", ".com")   = "www.domain"

//删除结尾匹配的字符串,找都不到返回原字符串
 StringUtils.removeEnd("www.domain.com", "domain") = "www.domain.com"

//忽略大小写的
StringUtils.removeEndIgnoreCase("www.domain.com", ".COM") = "www.domain")

//删除所有空白（好用）
StringUtils.deleteWhitespace("abc")        = "abc"
 StringUtils.deleteWhitespace("   ab  c  ") = "abc"
```

替换函数

replace(String text, String searchString, String replacement)

replace(String text, String searchString, String replacement, int max)

replaceChars(String str, char (煤焦) searchChar, char replaceChar)

replaceEach(String text, String[] searchList, String[] replacementList)

replaceEachRepeatedly(String text, String[] searchList, String[] replacementList)

replaceOnce(String text, String searchString, String replacement)

replacePattern(String source, String regex, String replacement)

overlay(String str,String overlay,int start,int end)

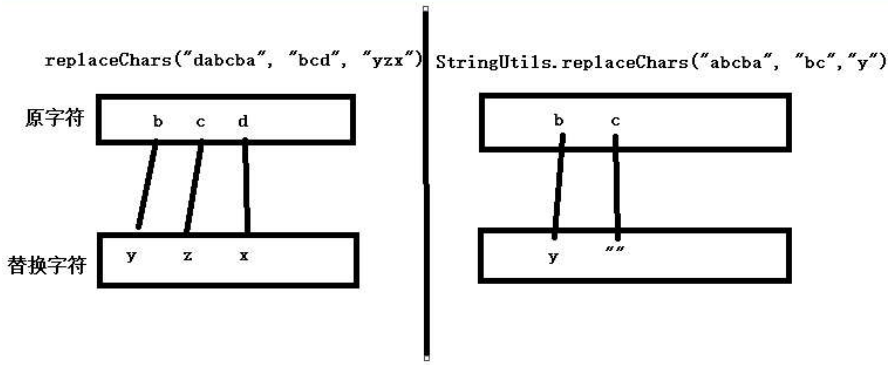
- 例子

```
StringUtils.replace("aba", "a", "")    = "b"
 StringUtils.replace("aba", "a", "z")   = "zbz"

//数字就是替换个数，0代表不替换，1代表从开始数起第一个，-1代表全部替换
StringUtils.replace("abaa", "a", "", -1)   = "b"
 StringUtils.replace("abaa", "a", "z", 0)   = "abaa"
 StringUtils.replace("abaa", "a", "z", 1)   = "zbaa"
 StringUtils.replace("abaa", "a", "z", 2)   = "zbza"
 StringUtils.replace("abaa", "a", "z", -1)  = "zbzz"
```

- **replaceEach是replace的增强版**，搜索列表和替换长度必须一致，否则报IllegalArgumentException异常：

```
StringUtils.replaceEach("abcde", new String[]{"ab", "d"}, new String[]{"w
  StringUtils.replaceEach("abcde", new String[]{"ab", "d"}, new String[]{
StringUtils.replaceChars("dabcba", "bcd", "yzx") = "xayzya"
StringUtils.replaceChars("abcba", "bc", "y")   = "ayya"
```



- replaceOnce只替换一次：

```
StringUtils.replaceOnce("aba", "a", "")    = "ba"
 StringUtils.replaceOnce("aba", "a", "z")   = "zba"
StringUtils.replaceEachRepeatedly("abcde", new String[]{"ab", "d"}, new S
  //这是一个非常奇怪的函数，本来自己料想结果应该是“dcte”的，可是结果居然是tcte
StringUtils.replaceEachRepeatedly("abcde", new String[]{"ab", "d"}, new S


StringUtils.overlay("abcdef", "zzzz", 2, 4)   = "abzzzzef"
 StringUtils.overlay("abcdef", "zzzz", 4, 2)   = "abzzzzef"
 StringUtils.overlay("abcdef", "zzzz", -1, 4)  = "zzzzef"
 StringUtils.overlay("abcdef", "zzzz", 2, 8)   = "abzzzz"
 StringUtils.overlay("abcdef", "zzzz", -2, -3) = "zzzzabcdef"
 StringUtils.overlay("abcdef", "zzzz", 8, 10)  = "abcdefzzzz"
```

反转函数

reverse(String str)

reverseDelimited(String str, char (煤焦) separatorChar)

例:

```
    StringUtils.reverse("bat") = "tab"
    StringUtils.reverseDelimited("a.b.c", 'x') = "a.b.c"
     StringUtils.reverseDelimited("a.b.c", ".") = "c.b.a"
```

分隔合并函数

split(String str)

split(String str, char (煤焦) separatorChar)

split(String str, String separatorChars)

split(String str, String separatorChars, int max)

splitByCharacterType(String str)

splitByCharacterTypeCamelCase(String str)

splitByWholeSeparator(String str, String separator (分隔符))

splitByWholeSeparator(String str, String separator (分隔符) , int max)

splitByWholeSeparatorPreserveAllTokens(String str, String separator (分隔符))

splitByWholeSeparatorPreserveAllTokens(String str, String separator (分隔符) , int max)

splitPreserveAllTokens(String str)

splitPreserveAllTokens(String str, char (煤焦) separatorChar)

splitPreserveAllTokens(String str, String separatorChars)

splitPreserveAllTokens(String str, String separatorChars, int max)

例:

```
    //用空白符做空格
    StringUtils.split("abc def")  = ["abc", "def"]

     StringUtils.split("abc  def") = ["abc", "def"]

    StringUtils.split("a..b.c", '.')   = ["a", "b", "c"]
     //用字符分割
    StringUtils.split("a:b:c", '.')    = ["a:b:c"]
    //0 或者负数代表没有限制
    StringUtils.split("ab:cd:ef", ":", 0)    = ["ab", "cd", "ef"]
     //分割字符串 ,可以设定得到数组的长度，限定为2
    StringUtils.split("ab:cd:ef", ":", 2)    = ["ab", "cd:ef"]
    //null也可以作为分隔
    StringUtils.splitByWholeSeparator("ab de fg", null)      = ["ab", "de", "
     StringUtils.splitByWholeSeparator("ab   de fg", null)    = ["ab", "de",
     StringUtils.splitByWholeSeparator("ab:cd:ef", ":")       = ["ab", "cd",
     StringUtils.splitByWholeSeparator("ab-!-cd-!-ef", "-!-") = ["ab", "cd",
    //带有限定长度的分隔
    StringUtils.splitByWholeSeparator("ab:cd:ef", ":", 2)       = ["ab", "cd:
```

join(byte[] array,char separator (分隔符))

join(Object[] array,char separator (分隔符))等方法

例:

```
//只有一个参数的join，简单合并在一起
StringUtils.join(["a", "b", "c"]) = "abc"
 StringUtils.join([null, "", "a"]) = "a"
//null的话，就是把字符合并在一起
StringUtils.join(["a", "b", "c"], null) = "abc"
//从index为0到3合并，注意是排除3的
StringUtils.join([null, "", "a"], ',', 0, 3)   = ",,a"
StringUtils.join(["a", "b", "c"], "--", 0, 3)  = "a--b--c"
//从index为1到3合并，注意是排除3的
 StringUtils.join(["a", "b", "c"], "--", 1, 3)  = "b--c"
 StringUtils.join(["a", "b", "c"], "--", 2, 3)  = "c"
```

截取函数

substring(String str,int start)

substringAfter(String str,String separator (分隔符))

substringBeforeLast(String str,String separator (分隔符))

substringAfterLast(String str,String separator (分隔符))

substringBetween(String str,String tag)

```
 StringUtils.substring("abcdefg", 0)  = "abcdefg"
 StringUtils.substring("abcdefg", 2)  = "cdefg"
 StringUtils.substring("abcdefg", 4)  = "efg"
 //start>0表示从左向右，start<0表示从右向左，start=0则从左第一位开始
 StringUtils.substring("abcdefg", -2) = "fg"
 StringUtils.substring("abcdefg", -4) = "defg"

 //从第二个参数字符串开始截取，排除第二个字符串
 StringUtils.substringAfter("abc", "a")   = "bc"
  StringUtils.substringAfter("abcba", "b") = "cba"
  StringUtils.substringAfter("abc", "c")   = ""

 //从最后一个字母出现开始截取
 StringUtils.substringBeforeLast("abcba", "b") = "abc"
 StringUtils.substringBeforeLast("abc", "c")   = "ab"
 StringUtils.substringBeforeLast("a", "a")     = ""
 StringUtils.substringBeforeLast("a", "z")     = "a"


 StringUtils.substringAfterLast("abc", "a")   = "bc"
  StringUtils.substringAfterLast("abcba", "b") = "a"
  StringUtils.substringAfterLast("abc", "c")   = ""

StringUtils.substringBetween("tagabctag", null)  = null
 StringUtils.substringBetween("tagabctag", "")    = ""
 StringUtils.substringBetween("tagabctag", "tag") = "abc"
```

截取分析图:

StringUtils.substring("abc", 2) = "c"
StringUtils.substring("abc", -2) = "bc"

```
// start>0&&end>0从左开始(包括左)到右结束(不包括右),
 //start<0&&end<0从右开始(包括右),再向左数到end结束(包括end)
StringUtils.substring("abc", -2, -1) = "b"
//这个我至今还没弄明白
 StringUtils.substring("abc", -4, 2)  = "ab"
```

相似度函数

一个字符串可以通过增加一个字符，删除一个字符，替换一个字符得到另外一个字符串，假设，我们把从字符串A转换成字符串B，前面3种操作所执行的最少次数称为AB相似度.

getLevenshteinDistance(CharSequence s, CharSequence t)
getLevenshteinDistance(CharSequence s, CharSequence t, int threshold)
StringUtils.getLevenshteinDistance(“elephant”, “ hippo (河马) ”) = 7
StringUtils.getLevenshteinDistance(“hippo”, “elephant”) = 7

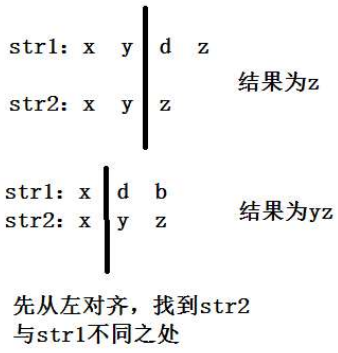
例:
//b替换为d
abc adc 度为 1
//abababababa去掉a，末尾加b
abababababa bababababab 度为 2
abcd acdb 度为2

差异函数

difference(String str1,String str2)

```
//在str1中寻找str2中没有的的字符串，并返回
StringUtils.difference("", "abc") = "abc"
StringUtils.difference("abc", "") = ""
StringUtils.difference("abc", "abc") = ""
StringUtils.difference("abc", "ab") = ""
StringUtils.difference("ab", "abxyz") = "xyz"
StringUtils.difference("abcde", "abxyz") = "xyz"
StringUtils.difference("abcde", "xyz") = "xyz"
```

图片理解：



缩短省略函数

abbreviate(String str, int maxWidth)

abbreviate(String str, int offset, int maxWidth)

abbreviateMiddle(String str, String middle, int length)

注意：

- 字符串的长度小于或等于最大长度，返回该字符串。
- 运算规律(substring(str, 0, max-3) + “…”)
- 如果最大长度小于4，则抛出异常。

```
    //(substring(str, 0, 6-3) + "...")
    StringUtils.abbreviate("abcdefg", 6) = "abc..."
    StringUtils.abbreviate("abcdefg", 7) = "abcdefg"
    StringUtils.abbreviate("abcdefg", 8) = "abcdefg"
    StringUtils.abbreviate("abcdefg", 4) = "a..."
    StringUtils.abbreviate("abcdefg", 3) = IllegalArgumentException
```

匹配计数函数

countMatches(CharSequence str,char ch)

```
    StringUtils.countMatches("abba", 0)  = 0
    StringUtils.countMatches("abba", 'a')   = 2
    StringUtils.countMatches("abba", 'b')  = 2
    StringUtils.countMatches("abba", 'x') = 0
```

删除空白函数

trim(String str)

deleteWhitespace(String str)

```
    StringUtils.trim("     ")       = ""
     StringUtils.trim("abc")         = "abc"
     StringUtils.trim("    abc    ") = "abc"
    //空的话，返回null
    StringUtils.trimToNull("     ")       = null
    StringUtils.trimToNull("abc")         = "abc"
    StringUtils.trimToNull("    abc    ") = "abc"
    StringUtils.trimToEmpty("     ")       = ""
     StringUtils.trimToEmpty("abc")         = "abc"
     StringUtils.trimToEmpty("    abc    ") = "abc"
```

-注意这两者的区别。

```
    StringUtils.deleteWhitespace("")           = ""
     StringUtils.deleteWhitespace("abc")        = "abc"
     StringUtils.deleteWhitespace("   ab  c  ") = "abc"
```

判断是否相等函数

equals(CharSequence cs1,CharSequence cs2)

equalsIgnoreCase(CharSequence str1, CharSequence str2)

```
    StringUtils.equals("abc", null)  = false
     StringUtils.equals("abc", "abc") = true
     StringUtils.equals("abc", "ABC") = false
    //忽略大小写
    StringUtils.equalsIgnoreCase("abc", null)  = false
     StringUtils.equalsIgnoreCase("abc", "abc") = true
     StringUtils.equalsIgnoreCase("abc", "ABC") = true
```

默认字符函数

defaultString(String str)

defaultString(String str,String defaultStr)

```
StringUtils.defaultString("")    = ""
 StringUtils.defaultString("bat") = "bat"
StringUtils.defaultString("", "NULL")    = ""
//如果第一个参数为空，这返回第二个默认参数
 StringUtils.defaultString("bat", "NULL") = "bat"
```

填充居中函数

leftPad/rightPad(String str,int size)

leftPad(String str,int size,char padChar)

center(String str,int size)

center(String str,int size,char padChar)

repeat(char ch,int repeat)

repeat(String str,String separator,int repeat)

appendIfMissing(String str, CharSequence suffix (后缀) , CharSequence… suffixes (后缀))

appendIfMissing(String str,CharSequence suffix,CharSequence… suffixes (后缀))

```
    StringUtils.leftPad("bat", 3)  = "bat"
    //左填充，默认填充空
    StringUtils.leftPad("bat", 5)  = "  bat"
```

```
//左填充，填充字符为z
StringUtils.leftPad("bat", 3, 'z')  = "bat"
 StringUtils.leftPad("bat", 5, 'z')  = "zzbat"
 StringUtils.leftPad("bat", 1, 'z')  = "bat"
 StringUtils.leftPad("bat", -1, 'z') = "bat"
//居中
StringUtils.center("ab", -1)  = "ab"
 StringUtils.center("ab", 4)   = " ab "
 StringUtils.center("abcd", 2) = "abcd"
 StringUtils.center("a", 4)    = " a
//居中，最后一个参数是填充字符或字符串
StringUtils.center("abcd", 2, ' ') = "abcd"
 StringUtils.center("a", 4, ' ')    = " a  "
 StringUtils.center("a", 4, 'y')    = "yayy"
//重复字符串，第二个参数是重复次数
StringUtils.repeat("a", 3)  = "aaa"
 StringUtils.repeat("ab", 2) = "abab"
 StringUtils.repeat("a", -2) = ""
//重复字符串，第二个参数是分割符，第三个参数是重复次数
StringUtils.repeat("", "x", 3)    = "xxx"
 StringUtils.repeat("?", ", ", 3)  = "?, ?, ?"

//
StringUtils.appendIfMissing("abcxyz", "xyz", "mno") = "abcxyz"
 StringUtils.appendIfMissing("abcmno", "xyz", "mno") = "abcmno"
 StringUtils.appendIfMissing("abcXYZ", "xyz", "mno") = "abcXYZxyz"
 StringUtils.appendIfMissing("abcMNO", "xyz", "mno") = "abcMNOxyz"
```

是否包含函数

containsOnly(CharSequence cs,char… valid)

containsNone(CharSequence cs,char… searchChars)

startsWith(CharSequence str,CharSequence prefix (前缀))

startsWithIgnoreCase(CharSequence str,CharSequence prefix (前缀))

startsWithAny(CharSequence string,CharSequence… searchStrings)

- 例子

```
//判断字符串中所有字符，是否都是出自参数2中
StringUtils.containsOnly("ab", "")      = false
StringUtils.containsOnly("abab", "abc") = true
StringUtils.containsOnly("ab1", "abc")  = false
StringUtils.containsOnly("abz", "abc")  = false

//判断字符串中所有字符，都不在参数2中。
StringUtils.containsNone("abab", 'xyz') = true
StringUtils.containsNone("ab1", 'xyz')  = true
StringUtils.containsNone("abz", 'xyz')  = false

//判断字符串是否以第二个参数开始
StringUtils.startsWith("abcdef", "abc") = true
 StringUtils.startsWith("ABCDEF", "abc") = false
```

索引下标函数

indexOf(CharSequence seq,CharSequence searchSeq)

indexOf(CharSequence seq,CharSequence searchSeq,int startPos)

indexOfIgnoreCase/lastIndexOfIgnoreCase(CharSequence str,CharSequence searchStr)

lastIndexOf(CharSequence seq,int searchChar)

```
    //返回第二个参数开始出现的索引值
    StringUtils.indexOf("aabaabaa", "a")  = 0
    StringUtils.indexOf("aabaabaa", "b")  = 2
    StringUtils.indexOf("aabaabaa", "ab") = 1

    //从第三个参数索引开始找起，返回第二个参数开始出现的索引值
    StringUtils.indexOf("aabaabaa", "a", 0)  = 0
     StringUtils.indexOf("aabaabaa", "b", 0)  = 2
     StringUtils.indexOf("aabaabaa", "ab", 0) = 1
     StringUtils.indexOf("aabaabaa", "b", 3)  = 5
     StringUtils.indexOf("aabaabaa", "b", 9)  = -1
    //返回第二个参数出现的最后一个索引值
    StringUtils.lastIndexOf("aabaabaa", 'a') = 7
     StringUtils.lastIndexOf("aabaabaa", 'b') = 5

    StringUtils.lastIndexOfIgnoreCase("aabaabaa", "A", 8)  = 7
    StringUtils.lastIndexOfIgnoreCase("aabaabaa", "B", 8)  = 5
    StringUtils.lastIndexOfIgnoreCase("aabaabaa", "AB", 8) = 4
    StringUtils.lastIndexOfIgnoreCase("aabaabaa", "B", 9)  = 5
```

读书感悟

来自《我们仨》

- 我一个人，怀念我们仨。
- 从今往后，咱们只有死别，再无生离.
- 我们这个家，很朴素；我们三个人，很单纯。我们与世无求，与人无争，只求相聚在一起，相守在一起，各自做力所能及的事。碰到困难，钟书总和我一同承当，困难就不复困难；还有个阿瑗相伴相助，不论什么苦涩艰辛的事，都能变得甜润。我们稍有一点快乐，也会变得非常快乐。所以我们仨是不寻常的遇合。
- 两年不见，她好像已经不认识了。她看见爸爸带回的行李放在妈妈床边，很不放心，猜疑地监视着，晚饭后，圆圆对爸爸发话了。
  “这是我的妈妈，你的妈妈在那边。”她要赶爸爸走。
  钟书很窝囊地笑说：“我倒问问你，是我先认识你妈妈，还是你先认识？”
  “自然我先认识，我一生出来就认识，你是长大了认识的。”
- 惟有身处卑微的人，最有机缘看到世态人情的真相。一个人不想攀高就不怕下跌，也不用倾轧排挤，可以保其天真，成其自然，潜心一志完成自己能做的事。

其他

如果有带给你一丝丝小快乐，就让快乐继续传递下去，欢迎转载，点赞，顶，欢迎留下宝贵的意见，多谢支持!

显示推荐内容

显示推荐内容

阿_毅 关注