Java String Format 示例

> 原文：https://dzone.com/articles/java-string-format-examples
>
> 本文内容来源于上面的地址，不完全按照原文进行翻译。
>
> 译文地址：http://blog.csdn.net/isea533/article/details/70193652

你是否经常忘记 Java String 格式说明符？或者你从来都没有时间学习过。以下是您可以使用的各种标识符参考。

你是否尝试阅读并理解 java 的 String 格式化文档？我觉得很难理解。尽管这个文档提供了所有的信息，但是它的组织结构让人失望。

String 格式化

在 Java 中格式化字符串最常用的方法就是 `String.format()`，如果有一个 Java 版本的 printf，它会像下面这样：

```
String output = String.format("%s = %d", "joe", 35);
```

对于控制台中的格式化输出，可以使用 System.out 或 System.err 中的 `printf()` 或者 `format()` 方法。

```
System.out.printf("My name is: %s%n", "joe");
```

创建一个 Formatter 并将其链接到 StringBuilder，通过`format()` 格式化输出的内容会追加到 StringBuilder 中。

```
StringBuilder sbuf = new StringBuilder();
Formatter fmt = new Formatter(sbuf);
fmt.format("PI = %f%n", Math.PI);
System.out.print(sbuf.toString());
// you can continue to append data to sbuf here.
```

格式说明符

以下是所有支持的转换说明符的快速参考。

说明符	适用于	输出
%a	浮点数 (除了BigDecimal)	浮点数的十六进制输出
%b	任何类型	如果为非空则为“true”，为空则为“false”
%c	字符	Unicode字符
%d	证书(包括byte, short, int, long, bigint)	十进制整数
%e	浮点数	科学计数的十进制数
%f	浮点数	十进制数
%g	浮点数	十进制数，根据值和精度可能以科学计数法显示
%h	任何类型	通过hashCode()方法输出的16进制数
%n	无	平台相关的换行符
%o	整数(包括byte, short, int, long, bigint)	八进制数
%s	任何类型	字符串
%t	日期/时间 (包含long, Calendar, Date和TemporalAccessor)	%t是日期/时间转换的前缀。后面还需要跟其他的标识，请参考下面的日期/时间转换。
%x	整数(包含byte, short, int, long, bigint)	十六进制字符串

日期和时间格式

注意：使用 "%T" 替换下面的 "%t" 可以将输出结果变成大写形式。

标识	注释
%tA	星期几的全名，例如 “`Sunday`“, “`Monday`“。
%ta	星期几的缩写，例如 “`Sun`“, “`Mon`“。
%tB	月份的全名，例如 “`January`“, “`February`“。
%tb	月份的缩写，例如 “`Jan`“, “`Feb`“。
%tC	年的世纪部分的格式为两位数，从 "`00`"到“`99`”。
%tc	日期和时间的格式为 “`%ta %tb %td %tT %tZ %tY`” 如 “`Fri Feb 17 07:45:42 PST 2017`“。
%tD	格式为 “`%tm/%td/%ty`“ 的日期。
%td	两位的日期格式，从 “`01`”到 “`31`“。
%te	没有前导0的日期，从 “1” 到 “31”。
%tF	使用 “`%tY-%tm-%td`“ 格式的 ISO 8601 日期。
%tH	24小时制的小时，从 “`00`” 到 “`23`“。
%th	同 %tb。
%tI	12小时制的小时，从 “`01`” 到 “`12`“。
%tj	带前导0的年中的日期，从 “`001`” 到“`366`“。
%tk	没有前导0的24小时制，从 “`0`” 到 “`23`“。
%tl	没有前导0的12小时制，从 “`1`” 到“`12`“。
%tM	带前导0的分钟，从 “`00`” 到“`59`“。
%tm	带前导0的月份，从 “`01`” 到 “`12`“。
%tN	带前导0的9位纳秒数，从 “000000000” to “999999999”.
%tp	和区域相关的 “am” or “pm” 标记。
%tQ	1970年1月1日00:00:00 UTC 以来的毫秒。
%tR	24小时制的时间，如：“`%tH:%tM`“。
%tr	12小时制的时间，如：“`%tI:%tM:%tS %Tp`“。
%tS	2位数字格式的秒，从 “00” 到 “60”。 “60” 需要支持闰秒。
%ts	1970年1月1日00:00:00 UTC以后的秒数。
%tT	24小时制的时分秒，如： “`%tH:%tM:%tS`“。
%tY	4位的年份格式，从 “`0000`” 到 “`9999`“。
%ty	2位的年份格式，从 “`00`” 到 “`99`“。

标识	注释
%tZ	时区缩写，如：“`UTC`“, “`PST`“。
%tz	与GMT的时区偏移量，如 `-0800`。

参数索引

参数索引是以 `$` 结尾，在 `%` 后面的数字，用于指定在参数列表中的参数。参数索引从 1 开始。

```
String.format("%2$s", 32, "Hello"); // 输出："Hello"
```

格式化整数

使用`%d`格式说明符，您可以使用所有整数类型的参数，包括byte，short，int，long 和BigInteger。默认格式：

```
String.format("%d", 93); // 输出：93
```

指定宽度：

```
String.format("|%20d|", 93); // 输出：|                  93|
```

指定宽度内的左对齐：

```
String.format("|%-20d|", 93); // 输出：|93                  |
```

用零填充：

```
String.format("|%020d|", 93); // 输出：|00000000000000000093|
```

用"+"号打印正数（负数总是包含"-"）：

```
String.format("|%+20d|", 93);  // 输出：|                 +93|
String.format("|%+20d|", -93); // 输出：|                 -93|
```

正数之前的空格，按正常值计算负数的“-”。

```
String.format("|% d|", 93);  // 输出：| 93|
String.format("|% d|", -36); // 输出：|-36|
```

使用和区域相关的千位分隔符，美国的是“,”：

```
String.format("|%,d|", 10000000); // 输出：|10,000,000|
```

中国的也一样：

```
String.format(Locale.CHINA, "|%,d|", 10000000)// 输出：|10,000,000|
```

使用左边括号括起来可以跳过“-”。

```
String.format("|%(d|", -36); // 输出：|(36)|
```

八进制输出。

```
String.format("|%o|"), 93); // 输出：135
```

十六进制输出。

```
String.format("|%x|", 93); // 输出：5d
```

八进制和十六进制输出的替代表示。

带前导0的八进制和带前导"`0x`"的十六进制输出：

```
String.format("|%#o|", 93); // 输出：0135
String.format("|%#x|", 93); // 输出：0x5d
String.format("|%#X|", 93); // 输出：0X5D
```

字符串和字符转换

默认格式。

打印整个字符串。

```
String.format("|%s|", "Hello World"); // 输出："Hello World"
```

指定字段长度。

```
String.format("|%20s|", "Hello World"); // 输出：|         Hello World|
```

左对齐文本。

```
String.format("|%-20s|", "Hello World"); // 输出：|Hello World         |
```

指定最大字符数。

```
String.format("|%.5s|", "Hello World"); // 输出：|Hello|
```

指定宽度和最大字符数。

```
String.format("|%20.5s|", "Hello World"); // 输出：|               Hello|
```

日期格式化

> 由于原文示例不涉及日期的例子，所以这一节是我个人增加的。

所有日期，除了纯数字显示的内容和区域无关外，文字显示的都和区域相关（下面示例的区域为中国）。

最简单的星期几。

```
String.format(Locale.US, "%tA", new Date()); // 输出：Sunday
String.format("%tA", new Date());            // 输出：星期日
```

"`%tc`"相当于“`%ta %tb %td %tT %tZ %tY`”，而且不需要自己指定索引。

```
String.format("%tc", new Date()); // 输出：星期日 四月 16 08:21:59 CST 2017
```

如果想要通过具体标识手动指定成上面的效果，需要增加索引。

```
String.format("%1$ta %1$tb %1$td %1$tT %1$tZ %1$tY", new Date());
// 输出：星期日 四月 16 08:24:06 CST 2017
```

常见的"`yyyy-MM-dd HH:mm:ss`"格式。

```
String.format("%1$tY-%1$tm-%1$td %1$tT", new Date());
// 输出：2017-04-16 08:31:16
```

总结

本篇教程解释了 Java 中的 String 格式化。我们涵盖了支持的格式说明符。数字和字符串格式都支持各种可代替的标识。