JAVA字符串格式化-String.format()的使用

常规类型的格式化

String类的format()方法用于创建格式化的字符串以及连接多个字符串对象。熟悉C语言的同学应该记得c语言的sprintf()方法，两者有类似之处。format()方法有两种重载形式。

format(String format, Object... args) 新字符串使用本地语言环境，制定字符串格式和参数生成格式化的新字符串。

format(Locale locale, String format, Object... args) 使用指定的语言环境，制定字符串格式和参数生成格式化的字符串。

显示不同转换符实现不同数据类型到字符串的转换，如图所示。

转 换 符	说　明	示　例
%s	字符串类型	"mingrisoft"
%c	字符类型	'm'
%b	布尔类型	true
%d	整数类型（十进制）	99
%x	整数类型（十六进制）	FF
%o	整数类型（八进制）	77
%f	浮点类型	99.99
%a	十六进制浮点类型	FF.35AE
%e	指数类型	9.38e+5
%g	通用浮点类型（f和e类型中较短的）	
%h	散列码	
%%	百分比类型	%
%n	换行符	
%tx	日期与时间类型（x代表不同的日期与时间转换符	

测试用例

```java
public static void main(String[] args) {
    String str=null;
    str=String.format("Hi,%s", "王力");
    System.out.println(str);
    str=String.format("Hi,%s:%s.%s", "王南","王力","王张");
```

```
        System.out.println(str);
        System.out.printf("字母a的大写是：%c %n", 'A');
        System.out.printf("3>7的结果是：%b %n", 3>7);
        System.out.printf("100的一半是：%d %n", 100/2);
        System.out.printf("100的16进制数是：%x %n", 100);
        System.out.printf("100的8进制数是：%o %n", 100);
        System.out.printf("50元的书打8.5折扣是：%f 元%n", 50*0.85);
        System.out.printf("上面价格的16进制数是：%a %n", 50*0.85);
        System.out.printf("上面价格的指数表示：%e %n", 50*0.85);
        System.out.printf("上面价格的指数和浮点数结果的长度较短的是：%g %n", 50*0.85);
        System.out.printf("上面的折扣是%d%% %n", 85);
        System.out.printf("字母A的散列码是：%h %n", 'A');
    }
```

输出结果

```
[plain]
Hi,王力
Hi,王南:王力.王张
字母a的大写是：A
3>7的结果是：false
100的一半是：50
100的16进制数是：64
100的8进制数是：144
50元的书打8.5折扣是：42.500000 元
上面价格的16进制数是：0x1.54p5
上面价格的指数表示：4.250000e+01
上面价格的指数和浮点数结果的长度较短的是：42.5000
上面的折扣是85%
字母A的散列码是：41
```

搭配转换符的标志，如图所示。

标　志	说　明	示　例	结　果
+	为正数或者负数添加符号	("%+d",15)	+15
–	左对齐	("%-5d",15)	|15　|
0	数字前面补0	("%04d", 99)	0099
空格	在整数之前添加指定数量的空格	("% 4d", 99)	|　99|
,	以“,”对数字分组	("%,f", 9999.99)	9,999.990000
(	使用括号包含负数	("%(f", -99.99)	(99.990000)
#	如果是浮点数则包含小数点，如果是16进制或8进制则添加0x或0	("%#x", 99) ("%#o", 99)	0x63 0143

<	格式化前一个转换符所描述的参数	("%f和%<3.2f", 99.45)	99.450000和99.45
$	被格式化的参数索引	("%1$d,%2$s", 99,"abc")	99,abc

测试用例

```java
public static void main(String[] args) {
    String str=null;
    //$使用
    str=String.format("格式参数$的使用：%1$d,%2$s", 99,"abc");
    System.out.println(str);
    //+使用
    System.out.printf("显示正负数的符号：%+d与%d%n", 99,-99);
    //补0使用
    System.out.printf("最牛的编号是：%03d%n", 7);
    //空格使用
    System.out.printf("Tab键的效果是：% 8d%n", 7);
    //.使用
    System.out.printf("整数分组的效果是：%,d%n", 9989997);
    //空格和小数点后面个数
    System.out.printf("一本书的价格是：% 50.5f元%n", 49.8);
}
```

输出结果

```plain
格式参数$的使用：99,abc
显示正负数的符号：+99与-99
最牛的编号是：007
Tab键的效果是：       7
整数分组的效果是：9,989,997
一本书的价格是：                                          49.80000元
```

日期和事件字符串格式化

在程序界面中经常需要显示时间和日期，但是其显示的 格式经常不尽人意，需要编写大量的代码经过各种算法才得到理想的日期与时间格式。字符串格式中还有%tx转换符没有详细介绍，它是专门用来格式化日期和时 间的。%tx转换符中的x代表另外的处理日期和时间格式的转换符，它们的组合能够将日期和时间格式化成多种格式。

常见日期和时间组合的格式，如图所示。

转 换 符	说　明	示　例
c	包括全部日期和时间信息	星期六 十月 27 14:21:20 CST 2007
F	“年-月-日”格式	2007-10-27

D	“月/日/年”格式	10/27/07
r	“HH:MM:SS PM”格式（12时制）	02:25:51 下午
T	“HH:MM:SS”格式（24时制）	14:28:16
R	“HH:MM”格式（24时制）	14:28

测试用例

```java
public static void main(String[] args) {
    Date date=new Date();
    //c的使用
    System.out.printf("全部日期和时间信息：%tc%n",date);
    //f的使用
    System.out.printf("年-月-日格式：%tF%n",date);
    //d的使用
    System.out.printf("月/日/年格式：%tD%n",date);
    //r的使用
    System.out.printf("HH:MM:SS PM格式（12时制）：%tr%n",date);
    //t的使用
    System.out.printf("HH:MM:SS格式（24时制）：%tT%n",date);
    //R的使用
    System.out.printf("HH:MM格式（24时制）：%tR",date);
}
```

输出结果

```plain
全部日期和时间信息：星期一 九月 10 10:43:36 CST 2012
年-月-日格式：2012-09-10
月/日/年格式：09/10/12
HH:MM:SS PM格式（12时制）：10:43:36 上午
HH:MM:SS格式（24时制）：10:43:36
HH:MM格式（24时制）：10:43
```

定义日期格式的转换符可以使日期通过指定的转换符生成新字符串。这些日期转换符如图所示。

```java
public static void main(String[] args) {
    Date date=new Date();
    //b的使用，月份简称
    String str=String.format(Locale.US,"英文月份简称：%tb",date);
    System.out.println(str);

    System.out.printf("本地月份简称：%tb%n",date);
    //B的使用，月份全称
    str=String.format(Locale.US,"英文月份全称：%tB",date);
    System.out.println(str);
    System.out.printf("本地月份全称：%tB%n",date);
    //a的使用，星期简称
```

```
        str=String.format(Locale.US,"英文星期的简称：%ta",date);
        System.out.println(str);
        //A的使用，星期全称
        System.out.printf("本地星期的简称：%tA%n",date);
        //C的使用，年前两位
        System.out.printf("年的前两位数字（不足两位前面补0）：%tC%n",date);
        //y的使用，年后两位
        System.out.printf("年的后两位数字（不足两位前面补0）：%ty%n",date);
        //j的使用，一年的天数
        System.out.printf("一年中的天数（即年的第几天）：%tj%n",date);
        //m的使用，月份
        System.out.printf("两位数字的月份（不足两位前面补0）：%tm%n",date);
        //d的使用，日（二位，不够补零）
        System.out.printf("两位数字的日（不足两位前面补0）：%td%n",date);
        //e的使用，日（一位不补零）
        System.out.printf("月份的日（前面不补0）：%te",date);
    }
```

输出结果

```
[plain]
英文月份简称：Sep
本地月份简称：九月
英文月份全称：September
本地月份全称：九月
英文星期的简称：Mon
本地星期的简称：星期一
年的前两位数字（不足两位前面补0）：20
年的后两位数字（不足两位前面补0）：12
一年中的天数（即年的第几天）：254
两位数字的月份（不足两位前面补0）：09
两位数字的日（不足两位前面补0）：10
月份的日（前面不补0）：10
```

和日期格式转换符相比，时间格式的转换符要更多、更精确。它可以将时间格式化成时、分、秒甚至时毫秒等单位。格式化时间字符串的转换符如图所示。

转 换 符	说　明	示　例
H	2位数字24时制的小时（不足2位前面补0）	15
I	2位数字12时制的小时（不足2位前面补0）	03
k	2位数字24时制的小时（前面不补0）	15
l	2位数字12时制的小时（前面不补0）	3
M	2位数字的分钟（不足2位前面补0）	03
S	2位数字的秒（不足2位前面补0）	09
L	3位数字的毫秒（不足3位前面补0）	015
N	9位数字的毫秒数（不足9位前面补0）	562000000
		中：下午

p	小写字母的上午或下午标记	英：pm
z	相对于GMT的RFC822时区的偏移量	+0800
Z	时区缩写字符串	CST

s	1970-1-1 00:00:00 到现在所经过的秒数	1193468128
Q	1970-1-1 00:00:00 到现在所经过的毫秒数	1193468128984

测试代码

```java
public static void main(String[] args) {
    Date date = new Date();
    //H的使用
    System.out.printf("2位数字24时制的小时（不足2位前面补0）:%tH%n", date);
    //I的使用
    System.out.printf("2位数字12时制的小时（不足2位前面补0）:%tI%n", date);
    //k的使用
    System.out.printf("2位数字24时制的小时（前面不补0）:%tk%n", date);
    //l的使用
    System.out.printf("2位数字12时制的小时（前面不补0）:%tl%n", date);
    //M的使用
    System.out.printf("2位数字的分钟（不足2位前面补0）:%tM%n", date);
    //S的使用
    System.out.printf("2位数字的秒（不足2位前面补0）:%tS%n", date);
    //L的使用
    System.out.printf("3位数字的毫秒（不足3位前面补0）:%tL%n", date);
    //N的使用
    System.out.printf("9位数字的毫秒数（不足9位前面补0）:%tN%n", date);
    //p的使用
    String str = String.format(Locale.US, "小写字母的上午或下午标记(英)：%tp", date);
    System.out.println(str);
    System.out.printf("小写字母的上午或下午标记（中）：%tp%n", date);
    //z的使用
    System.out.printf("相对于GMT的RFC822时区的偏移量:%tz%n", date);
    //Z的使用
    System.out.printf("时区缩写字符串:%tZ%n", date);
    //s的使用
    System.out.printf("1970-1-1 00:00:00 到现在所经过的秒数：%ts%n", date);
    //Q的使用
    System.out.printf("1970-1-1 00:00:00 到现在所经过的毫秒数：%tQ%n", date);
}
```

输出结果

```plain
2位数字24时制的小时（不足2位前面补0）:11
2位数字12时制的小时（不足2位前面补0）:11
2位数字24时制的小时（前面不补0）:11
```

```
2位数字12时制的小时（前面不补0）:11
2位数字的分钟（不足2位前面补0）:03
2位数字的秒（不足2位前面补0）:52
3位数字的毫秒（不足3位前面补0）:773
9位数字的毫秒数（不足9位前面补0）:773000000
小写字母的上午或下午标记(英)：am
小写字母的上午或下午标记（中）：上午
相对于GMT的RFC822时区的偏移量:+0800
时区缩写字符串:CST
1970-1-1 00:00:00 到现在所经过的秒数：1347246232
1970-1-1 00:00:00 到现在所经过的毫秒数：1347246232773
来源： http://blog.csdn.net/lonely_fireworks/article/details/7962171/
```