**JAVA字符串格式化-String.format()的使用**

常規類型的格式化

String類的format()方法用於創建格式化的字符串以及連接多個字符串對象。熟悉C語言的同學應該記得C語言的sprintf()方法，兩者有類似之處。format()方法有兩種重載形式。

format(String format, Object... args) 新字符串使用本地語言環境，制定字符串格式和參數生成格式化的新字符串。

format(Locale locale, String format, Object... args) 使用指定的語言環境，制定字符串格式和參數生成格式化的字符串。

顯示不同轉換符實現不同數據類型到字符串的轉換，如圖所示。

|  |  |  |
| --- | --- | --- |
| 轉 換 符 | 說 明 | 示 例 |
| %s | 字符串類型 | "mingrisoft" |
| %c | 字符類型 | 'm' |
| %b | 布爾類型 | true |
| %d | 整數類型（十進制） | 99 |
| %x | 整數類型（十六進制） | FF |
| %o | 整數類型（八進制） | 77 |
| %f | 浮點類型 | 99.99 |
| %a | 十六進制浮點類型 | FF.35AE |
| %e | 指數類型 | 9.38e+5 |
| %g | 通用浮點類型（f和e類型中較短的） |  |
| %h | 散列碼 |  |
| %% | 百分比類型 | ％ |
| %n | 換行符 |  |
| %tx | 日期與時間類型（x代表不同的日期與時間轉換符 |  |

測試用例

**[java]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. **public static void** main(String[] args) {
2. String str= **null**;
3. str=String.format( "Hi,%s" ,  "王力" );
4. System.out.println(str);
5. str=String.format( "Hi,%s:%s.%s" ,  "王南" , "王力" , "王張" );
6. System.out.println(str);
7. System.out.printf( "字母a的大寫是：%c %n" ,  'A' );
8. System.out.printf( "3>7的結果是：%b %n" ,  3 > 7 );
9. System.out.printf( "100的一半是：%d %n" ,  100 / 2 );
10. System.out.printf( "100的16進制數是：%x %n" ,  100 );
11. System.out.printf( "100的8進制數是：%o %n" ,  100 );
12. System.out.printf( "50元的書打8.5折扣是：%f元%n" ,  50 \* 0.85 );
13. System.out.printf( "上面價格的16進制數是：%a %n" ,  50 \* 0.85 );
14. System.out.printf( "上面價格的指數表示：%e %n" ,  50 \* 0.85 );
15. System.out.printf( "上面價格的指數和浮點數結果的長度較短的是：%g %n" ,  50 \* 0.85 );
16. System.out.printf( "上面的折扣是%d%% %n" ,  85 );
17. System.out.printf( "字母A的散列碼是：%h %n" ,  'A' );
18. }

輸出結果

**[plain]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. Hi,王力
2. Hi,王南:王力.王張
3. 字母a的大寫是：A
4. 3>7的結果是：false
5. 100的一半是：50
6. 100的16進制數是：64
7. 100的8進制數是：144
8. 50元的書打8.5折扣是：42.500000 元
9. 上面價格的16進制數是：0x1.54p5
10. 上面價格的指數表示：4.250000e+01
11. 上面價格的指數和浮點數結果的長度較短的是：42.5000
12. 上面的折扣是85%
13. 字母A的散列碼是：41

搭配轉換符的標誌，如圖所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 標 志 | 說 明 | 示 例 | 結 果 |
| + | 為正數或者負數添加符號 | ("%+d",15) | +15 |
| − | 左對齊 | ("%-5d",15) | |15 | |
| 0 | 數字前面補0 | ("%04d", 99) | 0099 |
| 空格 | 在整數之前添加指定數量的空格 | ("% 4d", 99) | | 99| |
| , | 以“,”對數字分組 | ("%,f", 9999.99) | 9,999.990000 |
| ( | 使用括號包含負數 | ("%(f", -99.99) | (99.990000) |
| # | 如果是浮點數則包含小數點，如果是16進製或8進制則添加0x或0 | ("%#x", 99)  ("%#o", 99) | 0x63  0143 |
| < | 格式化前一個轉換符所描述的參數 | ("%f和%<3.2f", 99.45) | 99.450000和99.45 |
| $ | 被格式化的參數索引 | ("%1$d,%2$s", 99,"abc") | 99,abc |

測試用例

**[java]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. **public static void** main(String[] args) {
2. String str= **null**;
3. //$使用
4. str=String.format( "格式參數$的使用：%1$d,%2$s" ,  99 , "abc" );
5. System.out.println(str);
6. //+使用
7. System.out.printf( "顯示正負數的符號：%+d與%d%n" ,  99 ,- 99 );
8. //補O使用
9. System.out.printf( "最牛的編號是：%03d%n" ,  7 );
10. //空格使用
11. System.out.printf( "Tab鍵的效果是：% 8d%n" ,  7 );
12. //.使用
13. System.out.printf( "整數分組的效果是：%,d%n" ,  9989997 );
14. //空格和小數點後面個數
15. System.out.printf( "一本書的價格是：% 50.5f元%n" ,  49.8 );
16. }

輸出結果

**[plain]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. 格式參數$的使用：99,abc
2. 顯示正負數的符號：+99與-99
3. 最牛的編號是：007
4. Tab鍵的效果是： 7
5. 整數分組的效果是：9,989,997
6. 一本書的價格是： 49.80000元

**日期和事件字符串格式化**

在程序界面中經常需要顯示時間和日期，但是其顯示的格式經常不盡人意，需要編寫大量的代碼經過各種算法才得到理想的日期與時間格式。字符串格式中還有%tx轉換符沒有詳細介紹，它是專門用來格式化日期和時間的。%tx轉換符中的x代表另外的處理日期和時間格式的轉換符，它們的組合能夠將日期和時間格式化成多種格式。

常見日期和時間組合的格式，如圖所示。

|  |  |  |
| --- | --- | --- |
| 轉 換 符 | 說 明 | 示 例 |
| c | 包括全部日期和時間信息 | 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時制） | 02:25:51 下午 |
| T | “HH:MM:SS”格式（24時制） | 14:28:16 |
| R | “HH:MM”格式（24時制） | 14:28 |

測試用例

**[java]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. **public static void** main(String[] args) {
2. Date date= **new** Date();
3. //c的使用
4. System.out.printf( "全部日期和時間信息：%tc%n" ,date);
5. //f的使用
6. System.out.printf( "年-月-日格式：%tF%n" ,date);
7. //d的使用
8. System.out.printf( "月/日/年格式：%tD%n" ,date);
9. //r的使用
10. System.out.printf( "HH:MM:SS PM格式（12時制）：%tr%n" ,date);
11. //t的使用
12. System.out.printf( "HH:MM:SS格式（24時制）：%tT%n" ,date);
13. //R的使用
14. System.out.printf( "HH:MM格式（24時制）：%tR" ,date);
15. }

輸出結果

**[plain]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. 全部日期和時間信息：星期一 九月 10 10:43:36 CST 2012
2. 年-月-日格式：2012-09-10
3. 月/日/年格式：09/10/12
4. HH:MM:SS PM格式（12時制）：10:43:36 上午
5. HH:MM:SS格式（24時制）：10:43:36
6. HH:MM格式（24時制）：10:43

定義日期格式的轉換符可以使日期通過指定的轉換符生成新字符串。這些日期轉換符如圖所示。

**[java]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. **public static void** main(String[] args) {
2. Date date= **new** Date();
3. //b的使用，月份簡稱
4. String str=String.format(Locale.US, "英文月份簡稱：%tb" ,date);
5. System.out.println(str);
6. System.out.printf( "本地月份簡稱：%tb%n" ,date);
7. //B的使用，月份全稱
8. str=String.format(Locale.US, "英文月份全稱：%tB" ,date);
9. System.out.println(str);
10. System.out.printf( "本地月份全稱：%tB%n" ,date);
11. //a的使用，星期簡稱
12. str=String.format(Locale.US, "英文星期的簡稱：%ta" ,date);
13. System.out.println(str);
14. //A的使用，星期全稱
15. System.out.printf( "本地星期的簡稱：%tA%n" ,date);
16. //C的使用，年前兩位
17. System.out.printf( "年的前兩位數字（不足兩位前面補0）：%tC%n" ,date);
18. //y的使用，年後兩位
19. System.out.printf( "年的後兩位數字（不足兩位前面補0）：%ty%n" ,date);
20. //j的使用，一年的天數
21. System.out.printf( "一年中的天數（即年的第幾天）：%tj%n" ,date);
22. //m的使用，月份
23. System.out.printf( "兩位數字的月份（不足兩位前面補0）：%tm%n" ,date);
24. //d的使用，日（二位，不夠補零）
25. System.out.printf( "兩位數字的日（不足兩位前面補0）：%td%n" ,date);
26. //e的使用，日（一位不補零）
27. System.out.printf( "月份的日（前面不補0）：%te" ,date);
28. }

輸出結果

**[plain]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. 英文月份簡稱：Sep
2. 本地月份簡稱：九月
3. 英文月份全稱：September
4. 本地月份全稱：九月
5. 英文星期的簡稱：Mon
6. 本地星期的簡稱：星期一
7. 年的前兩位數字（不足兩位前面補0）：20
8. 年的後兩位數字（不足兩位前面補0）：12
9. 一年中的天數（即年的第幾天）：254
10. 兩位數字的月份（不足兩位前面補0）：09
11. 兩位數字的日（不足兩位前面補0）：10
12. 月份的日（前面不補0）：10

和日期格式轉換符相比，時間格式的轉換符要更多、更精確。它可以將時間格式化成時、分、秒甚至時毫秒等單位。格式化時間字符串的轉換符如圖所示。

|  |  |  |
| --- | --- | --- |
| 轉 換 符 | 說 明 | 示 例 |
| H | 2位數字24時制的小時（不足2位前面補0） | 15 |
| I | 2位數字12時制的小時（不足2位前面補0） | 03 |
| k | 2位數字24時制的小時（前面不補0） | 15 |
| l | 2位數字12時制的小時（前面不補0） | 3 |
| M | 2位數字的分鐘（不足2位前面補0） | 03 |
| S | 2位數字的秒（不足2位前面補0） | 09 |
| L | 3位數字的毫秒（不足3位前面補0） | 015 |
| N | 9位數字的毫秒數（不足9位前面補0） | 562000000 |
| p | 小寫字母的上午或下午標記 | 中：下午  英：pm |
| z | 相對於GMT的RFC822時區的偏移量 | +0800 |
| Z | 時區縮寫字符串 | CST |

|  |  |  |
| --- | --- | --- |
| s | 1970-1-1 00:00:00 到現在所經過的秒數 | 1193468128 |
| Q | 1970-1-1 00:00:00 到現在所經過的毫秒數 | 1193468128984 |

測試代碼

**[java]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. **public static void** main(String[] args) {
2. Date date =  **new** Date();
3. //H的使用
4. System.out.printf( "2位數字24時制的小時（不足2位前面補0）:%tH%n" , date);
5. //I的使用
6. System.out.printf( "2位數字12時制的小時（不足2位前面補0）:%tI%n" , date);
7. //k的使用
8. System.out.printf( "2位數字24時制的小時（前面不補0）:%tk%n" , date);
9. //l的使用
10. System.out.printf( "2位數字12時制的小時（前面不補0）:%tl%n" , date);
11. //M的使用
12. System.out.printf( "2位數字的分鐘（不足2位前面補0）:%tM%n" , date);
13. //S的使用
14. System.out.printf( "2位數字的秒（不足2位前面補0）:%tS%n" , date);
15. //L的使用
16. System.out.printf( "3位數字的毫秒（不足3位前面補0）:%tL%n" , date);
17. //N的使用
18. System.out.printf( "9位數字的毫秒數（不足9位前面補0）:%tN%n" , date);
19. //p的使用
20. String str = String.format(Locale.US,  "小寫字母的上午或下午標記(英)：%tp" , date);
21. System.out.println(str);
22. System.out.printf( "小寫字母的上午或下午標記（中）：%tp%n" , date);
23. //z的使用
24. System.out.printf( "相對於GMT的RFC822時區的偏移量:%tz%n" , date);
25. //Z的使用
26. System.out.printf( "時區縮寫字符串:%tZ%n" , date);
27. //s的使用
28. System.out.printf( "1970-1-1 00:00:00到現在所經過的秒數：%ts%n" , date);
29. //Q的使用
30. System.out.printf( "1970-1-1 00:00:00到現在所經過的毫秒數：%tQ%n" , date);
31. }

輸出結果

**[plain]**[view plaincopy](http://blog.csdn.net/lonely_fireworks/article/details/7962171)

1. 2位數字24時制的小時（不足2位前面補0）:11
2. 2位數字12時制的小時（不足2位前面補0）:11
3. 2位數字24時制的小時（前面不補0）:11
4. 2位數字12時制的小時（前面不補0）:11
5. 2位數字的分鐘（不足2位前面補0）:03
6. 2位數字的秒（不足2位前面補0）:52
7. 3位數字的毫秒（不足3位前面補0）:773
8. 9位數字的毫秒數（不足9位前面補0）:773000000
9. 小寫字母的上午或下午標記(英)：am
10. 小寫字母的上午或下午標記（中）：上午
11. 相對於GMT的RFC822時區的偏移量:+0800
12. 時區縮寫字符串:CST
13. 1970-1-1 00:00:00 到現在所經過的秒數：1347246232
14. 1970-1-1 00:00:00 到現在所經過的毫秒數：1347246232773

# String.Format 大全

0、0的格式化

string.Format("{0:8D8}", 3)//第一个8表示空8个位置,后一个8表示用0填写最多8位数据

1、格式化貨幣（跟系統的環境有關，中文系統默認格式化人民幣，英文系統格式化美元）

string.Format("{0:C}",0.2) 结果为：￥0.20 （英文操作系统结果：$0.20）

默認格式化小數點後面保留兩位小數，如果需要保留一位或者更多，可以指定位數

string.Format("{0:C1}",23.15)

結果為：￥23.2 （截取會自動四捨五入）  
格式化多個Object實例

string.Format("市场价：{0:C}，优惠价{1:C}",23.15,19.82)

string.Format("{0:D3}",23); //结果为：023

string.Format("{0:D2}",1223); //结果为：1223，（精度说明符指示结果字符串中所需的最少数字个数。）

3、用分號隔開的數字，並指定小數點後的位數

string.Format("{0:N}", 14200); // 结果为：14,200.00 （默认为小数点后面两位）

string.Format("{0:N3}", 14200.2458); // 结果为：14,200.246 （自动四舍五入）

4、格式化百分比

string.Format("{0:P}", 0.24583); // 结果为：24.58% （默认保留百分的两位小数）

string.Format("{0:P1}", 0.24583);// 结果为：24.6% （自动四舍五入）

5、零佔位符和數字佔位符

string.Format("{0:0000.00}", 12394.039); // 结果为：12394.04

string.Format("{0:0000.00}", 194.039); // 结果为：0194.04

string.Format("{0:###.##}", 12394.039); // 结果为：12394.04

string.Format("{0:####.#}", 194.039); // 结果为：194

下面的這段說明比較難理解，多測試一下實際的應用就可以明白了。

**零佔位符**：  
如果格式化的值在格式字符串中出現“0”的位置有一個數字，則此數字被複製到結果字符串中。小數點前最左邊的“0”的位置和小數點後最右邊的“0”的位置確定總在結果字符串中出現的數字範圍。  
“00”說明符使得值被舍入到小數點前最近的數字，其中零位總被捨去。  
數字佔位符：  
如果格式化的值在格式字符串中出現“#”的位置有一個數字，則此數字被複製到結果字符串中。否則，結果字符串中的此位置不存儲任何值。   
請注意，如果“0”不是有效數字，此說明符永不顯示“0”字符，即使“0”是字符串中唯一的數字。如果“0”是所顯示的數字中的有效數字，則顯示“0”字符。   
“##”格式字符串使得值被舍入到小數點前最近的數字，其中零總被捨去。  
PS：空格佔位符

string.Format("{0,-50}", theObj);//格式化成50个字符，原字符左对齐，不足则补空格

string.Format("{0,50}", theObj);//格式化成50个字符，原字符右对齐，不足则补空格

6、日期格式化

string.Format("{0:d}",System.DateTime.Now);// 结果为：2009-3-20 （月份位置不是03）

string.Format("{0:D}",System.DateTime.Now);// 结果为：2009年3月20日

string.Format("{0:f}",System.DateTime.Now);// 结果为：2009年3月20日 15:37

string.Format("{0:F}",System.DateTime.Now);// 结果为：2009年3月20日 15:37:52

string.Format("{0:g}",System.DateTime.Now);// 结果为：2009-3-20 15:38

string.Format("{0:G}",System.DateTime.Now);// 结果为：2009-3-20 15:39:27

string.Format("{0:m}",System.DateTime.Now);// 结果为：3月20日

string.Format("{0:t}",System.DateTime.Now);// 结果为：15:41

string.Format("{0:T}",System.DateTime.Now);// 结果为：15:41:50

String.Format的多格式定義：  
這裡所謂的多格式是指一個格式項中可以定義1~3個格式參數，每種格式參數用分號（；）隔開。帶2個和3個格式參數的格式項所對應的值必須是數值類型的，這樣才能判斷是否為負數、正數、零。  
帶1個格式參數：  
//以科學計數法的格式輸出

double p1 = 1000000;

Response.Write(String.Format("{0:E2}", p1));

帶2個格式參數：  
/ *當格式項對應的值為非負數，則選擇第一種格式；值為負數則選第二種格式* /

double p1 = 10000;

double p2 = -2420.50;

Response.Write(String.Format("{0:#,###0.00;#,###0.000;}<BR>", p1));

Response.Write(String.Format("{0:#,###0.00;#,###0.000;}", p2));

帶3個格式參數：  
/\*當格式項對應的值為正數則選擇第一張格式；  
負數則為第二中格式；  
值等於零則為第三種格式\*/

1double p1 = 10000;

double p2 = -2420.50;

double p3 = 0.00;

Response.Write(String.Format("{0:#,###0.00;#,###0.000;#,###0.0000}<BR>", p1));

Response.Write(String.Format("{0:#,###0.00;#,###0.000;#,###0.0000}<BR>", p3));

Response.Write(String.Format("{0:#,###0.00;#,###0.000;#,###0.0000}", p2));

補充：  
{0:N2}中的N3,f3表示格式化之後數據的類型以及小數的位數。如：N2表示帶2個小數的數字；  
與此類似：  
N或者n表示數字  
F或者f表示固定點  
E或者e表示科學計數法  
D或者d表示十進制數  
X或者x表示十六進制  
G或者g表示常規  
C或者c表示貨幣

String.Format和WriteLine遵守同樣的格式化規則。格式化的格式如下：”{ N [, M ][: formatString ]}”, arg1, … argN，在這個格式中：

1） N是從0開始的整數，表示要格式化的參數的個數

2） M是一個可選的整數，表示格式化後的參數所佔的寬度，如果M是負數，那麼格式化後的值就是左對齊的，如果M是正數，那麼格式化後的值是右對齊的

3） formatString是另外一個可選的參數，表示格式代碼  
字母含義   
C或c Currency貨幣格式   
D或d Decimal十進制格式（十進制整數，不要和.Net的Decimal數據類型混淆了）   
E或e Exponent指數格式   
F或f Fixed point固定精度格式   
G或g General常用格式   
N或n用逗號分割千位的數字，比如1234將會被變成1,234   
P或p Percentage百分符號格式   
R或r Round-trip圓整（只用於浮點數）保證一個數字被轉化成字符串以後可以再被轉回成同樣的數字   
X或x Hex 16進制格式   
string str=string.Format(“{0,30:d}{1,10 :c}{2,6:c}{3,15:c}”,1234,12,1,10);的輸出是這樣的  
1234 ￥12.00 ￥1.00 ￥10.00