System Properties (系統參數)

System 類別 ➔取得 System.getProperty(String key) ·

設定 System.setProperty(String key · String val)

key	Description
java. version	Java Runtime Enviroment
java.home	JVM 所在目錄
java.vm.version	JVM 版本
java.compiler	JIT compiler 的名字
os.name	作業系統的名字
os.version	作業系統的版本
file.separator	Unix /
path.separator	Unix .
user.name	user name
user.home	user 的根目錄
user.dir	user 目前的工作目錄

```
public static void 系統參數() {

    System.out.println(System.getProperty("java.version")); //1.7.0_07
    System.out.println(System.getProperty("os.name")); //Windows 7
    System.out.println(System.getProperties());
    System.out.println(System.getProperties().get("os.name"));
    //System 提供的設定與取得
    System.setProperty("three", "3");
    System.out.println("three=" + System.getProperty("three")); //3
}
```

```
1.7.0_07
Windows 7
{java.runtime.name=Java(TM) SE Runtime Environment, sun.boot.library.path=C:\Program Files\Java\jdk1.7.0_07\jre\bin, java.vm.vers
, java.vm.specification.vendor=Oracle Corporation, user.variant=, os.name=Windows 7, sun.jnu.encoding=MS950, java.library.path=C:\
Windows 7
three=3
```

String · StringBuffer · StringBuilder 類別

1. String ➔ object 是不可變的 ➔產生之後值就不可被改變 · 因為不可被改變 · 所以 String 內大部份的方法都會 return 一個新的 String object · 經常建立新的 object 對系統是個大負擔
2. StringBuffer(方法都是 同步的) ➔為了不影響系統的 performance 若字串內的值經常需要被改就當用 StringBuffer 不要用 String
   ➔ 適合多執行緒環境
3. StringBuilder(方法都是 非同步的)➔方法都與 StringBuffer 一樣 · 5.0 以後才被發展出來 · 速度最快
   ➔ 適合單一執行緒環境

建立 String 的方式

1. 使用 String s = new String("abc");
2. 使用 String s = "abc";
3. 使用 operators "+"或"+=" 或 concat➔ String s = "abc" + "xyz" 或 String s = "abc".concat("xyz") // s = "abcxyz"
4. 使用 toString()

   Object class 所定義的 "toString()" method 其傳回值為 "getClsss().getName() + '@' + Integer. toHexString(hashCode())

   一個 自訂 class 可 override 這個 method 讓它的傳回值更有代表意義。

   8 個 wrapper c1asses 、StringBuffer 、StringBuilder 、Date 、File 等等 都有 Overrides 這個 toString()的方法
5. 使用 String s = String.valueOf(3 .14159); // s 的 content 為 "3.14159"

```java
public static void 三種字串型別的不同() {
    String x = "abcde";
    x.concat("xyz"); //產生新字串
    System.out.println("String 無重新指派 x=" + x);  //abcde

    x = x.concat("xyz"); //產生新字串 , 重新指派到新字串
    System.out.println("String 有重新指派 x=" + x); //abcdexyz

    StringBuffer y = new StringBuffer("abcde");
    y.append("xyz");
    System.out.println("StringBuffer y=" + y);  //abcdexyz

    StringBuilder z = new StringBuilder("abcde");
    z.append("xyz");
    System.out.println("StringBuilder z=" + z);  //abcdexyz
}
```

```
String 無重新指派 x=abcde
String 有重新指派 x=abcdexyz
StringBuffer y=abcdexyz
StringBuilder z=abcdexyz
```

```java
public class UtilityDemo1 {

    public int x;

    public UtilityDemo1(int x) {
        this.x = x;
    }
    public int hashCode() {
        return x;
    }
    public String toString() {
        return String.valueOf(x);
    }
    public boolean equals(Object o) {
        if (o != null && o instanceof UtilityDemo1) {
            if (x == ((UtilityDemo1) o).x) {
                return true;
            }
        }
        return false;
    }
}
```

```java
public static void equals已覆寫() {
    //字串 已覆寫
    String a = "abc";
    String b = "abc";
    System.out.println("a.equals(b)=" + a.equals(b));  //true

    //八個包裝類別 已覆寫
    Integer x = new Integer(4);
    Integer y = new Integer(4);
    System.out.println("x.equals(y) =" + x.equals(y));  //true

    // StringBuffer , StringBuilder 沒覆寫
    StringBuffer e = new StringBuffer("abc");
    StringBuffer f = new StringBuffer("abc");
    System.out.println("e.equals(f)=" + e.equals(f));  //false
    System.out.println("e.toString().equals(f.toString())=" + e.toString().equals(f.toString()));  //true

    //自訂類別 已覆寫
    UtilityDemo1 c = new UtilityDemo1(10);
    UtilityDemo1 d = new UtilityDemo1(10);
    System.out.println("c.equals(d)=" + c.equals(d));  //true
}
```

```java
public static void toString已覆寫() {
    //八個包裝類別 已覆寫
    Integer a = new Integer(4);
    System.out.println("a=" + a);  // 4
    //日期 已覆寫
    Date b = new Date();
    System.out.println("b=" + b);   //Mon Nov 13 09:52:11 CST 2017
    //StringBuffer , StringBuilder 已覆寫
    StringBuffer c = new StringBuffer("abc");
    System.out.println("c=" + c); //abc
    StringBuilder d = new StringBuilder("abc");
    System.out.println("d=" + d); //abc
    //集合 已覆寫
    Set e = new HashSet();
    e.add("a");
    e.add("b");
    e.add("c");
    System.out.println("e=" + e); //[a, b, c]
    //自訂類別 已覆寫
    UtilityDemo1 f = new UtilityDemo1(10);
    System.out.println("f=" + f);  // 10
}
```

字串的函數中若有起始值與終止值➔記住，都不包含終止值

```
public static void String的函數1() {
    // 字串的 三種型別都有 subString() 的方法
    // 但 subString==>會回傳 String 型別
    String s1 = "abcdefg";
    System.out.println("s1.substring(2)=" + s1.substring(2));  //cdefg
    System.out.println("s1.substring(2, 4)=" + s1.substring(2, 4)); //cd

    StringBuffer s2 = new StringBuffer("abcdefg");
    String s3 = s2.substring(2);
    System.out.println("s3=" + s3);

    StringBuilder s4 = new StringBuilder("abcdefg");
    String s5 = s4.substring(2);
    System.out.println("s5=" + s5);
}
```

```
cdefg
cd
```

```
public static void String的函數2() {
    //charAt
    String s1 = "abcdefg";
    System.out.println("s1.charAt(0) = " + s1.charAt(0)); //a

    //toCharArray
    char[] ar1;
    ar1 = "abcde".toCharArray();
    for (char x : ar1) {
        System.out.print(x + " ");
    }
    System.out.println();

    //getChars
    char[] ca = new char[5];  //空
    "abcdef".getChars(1, 4, ca, 1);
    for (char x : ca) {
        System.out.print(x + " ");
    }
}
```

```
a
a b c d e
  b c d
```

```
public static void String的函數3() {
    // startsWith
    System.out.println("abc123abc123".startsWith("abc"));  //true
    System.out.println("abc123abc123".endsWith("c123"));  //true
    System.out.println("abc123abc123".startsWith("3abc1", 5));  //true

    //compareTo
    System.out.println("abc".compareTo("abc"));  //0
    System.out.println("abc".compareTo("abcdef"));  //-3
    System.out.println("a".compareTo("Azzzz"));  //32

    //indexOf
    System.out.println("abc123abc123".indexOf("c12"));  //2
    System.out.println("abc123abc123".indexOf("c12", 3));  //8
    System.out.println("abc123abc123".lastIndexOf("c12", 8));  //8
    System.out.println("abc123abc123".lastIndexOf("c12", 7));  //2
}
```

```java
public static void String的函數4() {
    //replace
    System.out.println("abc123abc123".replace('a', 'x')); //xbc123xbc123"  1.4
    System.out.println("abc123abc123".replace("abc", "xyz"));// xyz123xyz123 5.0
    System.out.println("  abc 123   ".trim());  //abc 123
}
```

```java
public static void StringBuffer的函數() {
    StringBuffer sb = new StringBuffer("12345678");
    StringBuffer sc = new StringBuffer("12345678");

    System.out.println("sc.equals(sb) =" + sc.equals(sb)); //false , 沒有覆寫 equals
    System.out.println(sc.toString().equals(sb.toString())); //true

    String a = "abcdefg";
    String b = "abcdefg";

    System.out.println("a.equals(b) = " + a.equals(b)); //true 有覆寫 equals

    // "12345678"
    sb.setCharAt(3, 'a');
    System.out.println("sb = " + sb);  //123a5678
    //replace
    sb.replace(1, 6, "xyz");
    System.out.println("sb= " + sb);  //1xyz78
    //append
    sb.append("z");
    System.out.println("sb= " + sb);  //1xyz78z
    sb.append("3.1416");
    System.out.println("sb= " + sb);  //1xyz78z3.1416
    //setLength
    sb.setLength(3);
    System.out.println("sb= " + sb);  //1xy
    //delete
    sb.delete(1, 2);
    System.out.println("sb= " + sb);  //1y
    sb.append("wxyz"); //1ywxyz
    //insert
    sb.insert(2, "abc");
    System.out.println("sb= " + sb);  //1yabcwxyz
    //reverse
    sb.reverse();
    System.out.println("sb= " + sb);  //zyxwcbay1
}
```

尋找東西

當你遇到一大堆要處理的文字，或是在做某種螢幕抓取的動作，或許需要從檔內讀資料，都需要一個很容易的方法來從這堆文字中找到資料可以使用 java.util.Pattern，java.util.regex.Matcher 和 java.util.Scanner

符號化(剖析字串)

有加上某種分隔符號的檔案，想從這樣的檔案得有用的資料，用 String.split()的使用方法 和 java.util.Scanner 類別來符號化資料

格式化東西

要產生一個報表，而你需要將這樣的浮點變數 32500.000f 轉變成 $32，500.00 的字串可以使用 java.util.Formatter 類別和 printf()，及 format()函式

簡單的搜尋➔regex 引擎

```java
public static void 樣式匹配搜尋1() {

    Pattern p = Pattern.compile("ab");  //樣版
    Matcher m = p.matcher("abaaaba");
    boolean b = false;
    System.out.println("Pattern is " + m.pattern()); //ab

    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
Pattern is ab
在位置 0 找到 ab
在位置 4 找到 ab
```

```java
public static void 樣式匹配搜尋2() {

    Pattern p = Pattern.compile("aba");  //樣版
    Matcher m = p.matcher("abababa");  //索引 2 的 ab 被消耗了

    boolean b = false;
    System.out.println("Pattern is " + m.pattern());
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
Pattern is aba
在位置 0 找到 aba
在位置 4 找到 aba
```

使用中介字元

\d ➔一個數字， \s ➔一個空白字元，\w➔一個字的字元 (字母，數字，或 "_" 底線)

[abc]，[a-f]，[a-FA-F]

當中介字元 和 字串衝突的時候

String pattern= "\d" (沒有 \d 這樣的跳脫序列) ➔ \\d ➔第一個 反斜線告訴編譯器，不論下一個遇到的是什麼，都把它當 作普通字串， 不是一個跳脫序列

```java
public static void 樣式匹配搜尋3() {
    String x = "\\d";
    String y = "a12c3e456f";

    Pattern p = Pattern.compile(x);  //樣版
    Matcher m = p.matcher(y);

    boolean b = false;
    System.out.println("Pattern is " + m.pattern());
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
Pattern is \d
在位置 1 找到 1
在位置 2 找到 2
在位置 4 找到 3
在位置 6 找到 4
在位置 7 找到 5
在位置 8 找到 6
```

```java
public static void 樣式匹配搜尋4() {
    String x = "\\w";
    String y = "a 1 56 _Z";
    Pattern p = Pattern.compile(x);  //樣版

    Matcher m = p.matcher(y);
    boolean b = false;
    System.out.println("Pattern is " + m.pattern());
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
Pattern is \w
在位置 0 找到 a
在位置 2 找到 1
在位置 4 找到 5
在位置 5 找到 6
在位置 7 找到 _
在位置 8 找到 Z
```

```java
public static void 樣式匹配搜尋5() {
    String x = "\\d\\w";
    String y = "ab4 56_7ab";
    Pattern p = Pattern.compile(x);  //樣版

    Matcher m = p.matcher(y);
    boolean b = false;
    System.out.println("Pattern is " + m.pattern());
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
Pattern is \d\w
在位置 4 找到 56
在位置 7 找到 7a
```

```java
public static void 樣式匹配搜尋6() {
    String x = "[a-f]";
    String y = "ab4 56_7ab";
    Pattern p = Pattern.compile(x);  //樣版

    Matcher m = p.matcher(y);
    boolean b = false;
    System.out.println("Pattern is " + m.pattern());
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
Pattern is [a-f]
在位置 0 找到 a
在位置 1 找到 b
在位置 8 找到 a
在位置 9 找到 b
```

```java
public static void 樣式匹配搜尋7() {
    String x = "[abc]";
    String y = "ab4 5c6_7acb";
    Pattern p = Pattern.compile(x);  //樣版

    Matcher m = p.matcher(y);
    boolean b = false;
    System.out.println("Pattern is " + m.pattern());
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
Pattern is [abc]
在位置 0 找到 a
在位置 1 找到 b
在位置 5 找到 c
在位置 9 找到 a
在位置 10 找到 c
在位置 11 找到 b
```

```java
public static void 樣式匹配搜尋8() {
    String x = "[a-fA-F]";
    String y = "ab4 56_7ab9BF";
    Pattern p = Pattern.compile(x);  //樣版

    Matcher m = p.matcher(y);
    boolean b = false;
    System.out.println("Pattern is " + m.pattern());
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
Pattern is [a-fA-F]
在位置 0 找到 a
在位置 1 找到 b
在位置 8 找到 a
在位置 9 找到 b
在位置 11 找到 B
在位置 12 找到 F
```

使用量詞（Quantifier）來搜尋

* ➔0 次或多次 ? ➔ 0 次或 1 次 +➔1 次或多次

```java
public static void 樣式匹配搜尋9() {
    String x = "\\d";
    String y = "ab12cde345fg";
    Pattern p = Pattern.compile(x);
    Matcher m = p.matcher(y);
    boolean b = false;
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
在位置 2 找到 1
在位置 3 找到 2
在位置 7 找到 3
在位置 8 找到 4
在位置 9 找到 5
```

```java
public static void 樣式匹配搜尋10() {
    String x = "\\d+";
    String y = "ab12cde345fg";
    Pattern p = Pattern.compile(x);
    Matcher m = p.matcher(y);
    boolean b = false;
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
在位置 2 找到 12
在位置 7 找到 345
```

```java
public static void 樣式匹配搜尋11() {
    String x = "\\d*";
    String y = "ab12cde345fg";
    Pattern p = Pattern.compile(x);
    Matcher m = p.matcher(y);
    boolean b = false;
    while (b = m.find()) {
        System.out.println("在位置 " + m.start() + " " + "找到 " + m.group());
    }
}
```

```
在位置 0 找到
在位置 1 找到
在位置 2 找到 12
在位置 4 找到
在位置 5 找到
在位置 6 找到
在位置 7 找到 345
在位置 10 找到
在位置 11 找到
在位置 12 找到
```

符號化

1. 使用 String.split()進行符號化
2. 使用 Scanner 進行符號化

```java
public static void split1() {
    String x = "abeeeeee#cdy#efd";
    String y = "#";
    String[] z;
    z = x.split(y);
    for (String m : z) {
        System.out.println(m);
    }
}
```

```
abeeeeee
cdy
efd
```

```java
public static void split2() {
    String x = "abeeeeee5cdy67efd";
    String y = "\\d";
    String[] z;
    z = x.split(y);
    for (String m : z) {
        System.out.println(m);
    }
}
```

```
abeeeeee
cdy

efd
```

```java
public static void split3() {
    String x = "abeeeeee5cdy67efd";
    String y = "\\d+";
    String[] z;
    z = x.split(y);
    for (String m : z) {
        System.out.println(m);
    }
}
```

```
abeeeeee
cdy
efd
```

```java
public static void scanner1() {

    boolean x, y;
    int i;
    String s;

    String a = "1 true 34 hi";  // Scanner 會以空白分隔

    String b = "1 a 3 4";  // Scanner 會以空白分隔

    // Scanner s1 = new Scanner(a).useDelimiter(" ");
    Scanner s1 = new Scanner(a);

    while (x = s1.hasNext()) {
        s = s1.next();
        System.out.println(s + " ");
    }

    System.out.println();

    Scanner s2 = new Scanner(b);
    while (y = s2.hasNext()) {
        i = s2.nextInt();

        System.out.println(i + " "); //印完 1 後 會當掉--------
    }
}
```

```
1
true
34
hi

1
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:909)
        at java.util.Scanner.next(Scanner.java:1530)
        at java.util.Scanner.nextInt(Scanner.java:2160)
        at java.util.Scanner.nextInt(Scanner.java:2119)
        at java24_尋找與剖析符號化與格式化.Test24.scanner3(Test24.java:292)
        at java24_尋找與剖析符號化與格式化.Main.main(Main.java:24)
Java Result: 1
```

```java
public static void scanner2() {
    boolean x, y;
    int i, m;
    String s;
    String a = "1 true 34 hi";  // Scanner 會以空白分隔

    Scanner s1 = new Scanner(a);

    while (x = s1.hasNextInt()) { //剛好第一個是數字
        m = s1.nextInt();
        System.out.println(m);  //不會當掉 但只印出 1
    }
}
```

```
1
```

```java
public static void scanner3() {
    boolean x, y;
    int i, m;
    String s;
    String a = "a true 34 hi";  // Scanner 會以空白分隔

    Scanner s1 = new Scanner(a);
    while (x = s1.hasNextInt()) { //如果第一個遇到不是數字
        m = s1.nextInt();
        System.out.println(m);  //沒有任何輸出
    }
}
```

無任何輸出

```java
public static void scanner4() {
    boolean b2, b;
    int i;
    String s, hits = "";

    String a = "1 true 34 hi";  // Scanner 會以空白分隔

    Scanner s1 = new Scanner(a);
    Scanner s2 = new Scanner(a);

    while (s1.hasNext()) {
        s = s1.next();
        hits += "s";   //ssss
    }
    System.out.println("結果: " + hits);

    hits = "";
    while (b = s2.hasNext()) { //不管遇到什麼都會往前

        if (s2.hasNextInt()) {
            i = s2.nextInt();
            hits += "i";
        } else if (s2.hasNextBoolean()) {
            b2 = s2.nextBoolean();
            hits += "b";
        } else {
            s2.next();
            hits += "s2";
        }
    }
    System.out.println("結果: " + hits);  //ibis2
}
```

結果: ssss
結果: ibis2

```java
public static void scanner5() {
    String a = "Sue,5, true  ,3, abc, false";
    Scanner s1 = new Scanner(a).useDelimiter("\\s*,\\s*");
    boolean ss;

    while (s1.hasNext()) {

        if (s1.hasNextBoolean()) {
            ss = s1.nextBoolean();
            System.out.println(ss);
        } else {
            s1.next();
        }
    }
}
```

```
true
false
```

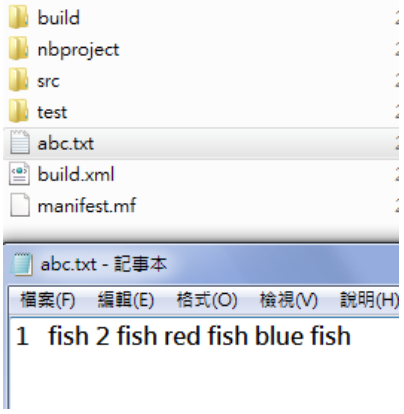
```java
public static void scanner6() {
    boolean x;
    String y;

    String input = "ab    ,    cd  ,   ef";
    Scanner s = new Scanner(input).useDelimiter("\\s*,\\s*");
    //  Scanner s = new Scanner(input).useDelimiter(",");
    while (x = s.hasNext()) {
        y = s.next();
        System.out.println(y);
    }
}
```

```
ab
cd
ef
```

```java
public static void scanner7() {
    boolean x;
    String y;

    String input = "1   fish 2 fish red fish blue fish";
    Scanner s = new Scanner(input).useDelimiter("\\s*fish\\s*");
    //Scanner s = new Scanner(input).useDelimiter("fish");

    while (x = s.hasNext()) {
        y = s.next();
        System.out.println(y);
    }
}
```

```
1
2
red
blue
```

```java
public static void scanner81() {
    boolean x;
    String y;
    Scanner s = null;

    try {
        File file = new File("abc.txt");
        s = new Scanner(file).useDelimiter("\\s*fish\\s*");
    } catch (FileNotFoundException e) {
        System.out.println("沒有發現檔案");
    }

    while (x = s.hasNext()) {
        y = s.next();
        System.out.println(y);
    }
}
```




```
1
2
red
blue
```

```java
public static void scanner9() {
    Scanner sc = new Scanner("123 A 3b c,45, x5x,76 82 L");
    do {
        if (sc.hasNextInt()) {
            System.out.print(sc.nextInt() + " ");
        }
    } while (sc.hasNext());  //do迴圈一碰到 非數字就不會繼續往前
    //只會印出 123 後就無窮迴圈了
}
```

```
123
```

java 字符串分解 StringTokenizer 用法（比 split()方法效率高）

1. StringTokenizer 能讓一個字串分解為一個一個的 單詞 或 標記
   <1>.int countTokens（）：返回 nextToken 方法被調用的次數。
   <2>.boolean hasMoreTokens（）：回傳是否還有分隔符。
   <3>.boolean hasMoreElements（）：回傳是否還有分隔符。
   <4>.String nextToken（）：回傳從當前位置到下一個分隔符的字串。
   <5>.Object nextElement（）：回傳從當前位置到下一個分隔符的字串。
   <6>.String nextToken（String delim）：與 4 類似，以指定的分隔符回傳結果。
2. StringTokenizer 的三個方法(建構函數)：
   <1>.StringTokenizer(String str)。默認以" \t\n\r\f"（空格，tab，換行，換頁）為分割符
   <2>.StringTokenizer(String str， String delim)。指定 delim 為分割符
   <3>.StringTokenizer(String str， String delim， boolean returnDelims)。returnDelims 為 true 的則 delim 分割符也為標記

```java
public static void StringTokenizer1() {
    StringTokenizer st = new StringTokenizer("www.ooobj.com", ".b");
    while (st.hasMoreTokens()) {
        System.out.println("Token:" + st.nextToken());
    }
}
```

```
Token:www
Token:ooo
Token:j
Token:com
```

```java
public static void StringTokenizer2() {
    StringTokenizer st = new StringTokenizer("www ooobj com");
    while (st.hasMoreElements()) {
        System.out.println("Token:" + st.nextToken());
    }
}
```

```
Token:www
Token:ooobj
Token:com
```

```java
public static void StringTokenizer3() {
    StringTokenizer st = new StringTokenizer("www.ooobj.com", ".", true);
    while (st.hasMoreElements()) {
        System.out.println("Token:" + st.nextToken());
    }
}
```

```
Token:www
Token:.
Token:ooobj
Token:.
Token:com
```

```java
public static void StringTokenizer4() {
    StringTokenizer st = new StringTokenizer("www.ooobj.com");
    while (st.hasMoreTokens()) {
        System.out.println("Token:" + st.nextToken("."));
    }
}
```

```
Token:www
Token:ooobj
Token:com
```

使用 printf() 和 format() 進行格式化

1. format()和 printf() 函式在 java5 被加入 java.io.PrintStream 內

   printf(“格式化字串” ,” 引數”)

2. %[引數索引$][標記][寬度][.精確度]轉換字元

引數索引 ➔一個整數，後面接著 $，表示哪一個引數應該在這個位置被印出來

標記 ➔雖然可以使用的標記有很多，但有下面幾個會被涵蓋在考題內

“-“：將引數向左靠齊， “+” ：包含引數的正負號， “0” ：用 0 將空白填滿， “,” ：使用地域特殊的群組分隔符號

“(“：將負數放在刮號內

寬度 ➔這值表示最少要被印出來的字元數目

精確度 ➔小數點後可以擺幾個數字

轉換字元➔ b 布林值 c 字元 d 整數 f 浮點數 s 字串

```java
public static void 數字的格式化1() {
    //   %[引數索引$][標記][寬度][精確度]轉換字元]
    int i1 = -123;
    int i2 = 12345;

    System.out.printf(">%1$(7d< \n", i1); //負數放在括號內 7碼 >  (123)<

    System.out.printf(">%0,7d< \n", i2); //用 0 填滿 , 7碼 >012,345<

    System.out.format(">%+-7d< \n", i2); //正負號 , 靠左對齊 7 碼 >+12345 <

    System.out.printf(">%2$b + %1$5d< \n", i1, false); //第 2 個數先顯示 布林
    //再顯示第1個數 整數  >false +  -123<
}
```

```
>  (123)<
>012,345<
>+12345 <
>false +  -123<
```

```java
public static void 數字的格式化2() {
    // System.out.printf("%f", 12); //會當掉
    // System.out.printf("%d", 12.5); //會當掉
    System.out.printf("%7.2f", 12.5);
    System.out.println();
    System.out.printf("%b", 10.2); //true
    System.out.println();
}
```

```
  12.50
true
```

```java
public static void 數字的格式化3() {

    DecimalFormat formatter = new DecimalFormat("#,###,###.00");

    System.out.printf("%s\n", formatter.format(1234567));

}
```

```
1,234,567.00
```

Java ResourceBundle 多語系(本地化)

Api：Locale

Api：ResourceBundle

這個類提供軟體國際化的捷徑。通過此類，可以使您所編寫的程式可以：

輕鬆地當地語系化或翻譯成不同的語言

一次處理多個語言環境

以後可以輕鬆地進行修改，支援更多的語言環境

這個類的作用就是讀取資源屬性檔（properties），然後根據.properties 檔的名稱資訊（當地語系化資訊），匹配當前系統的國別語言資訊（也可以程式指定），然後獲取相應的 properties 檔的內容。

使用這個類，要注意的一點是，這個 properties 檔的名字是有規範的：一般的命名規範是：自訂名_語言代碼_國別代碼.properties，如果是默認的，直接寫為：自訂名.properties

例如：

example.properties➔沒有任何語系與國別的資源檔時 會讀到

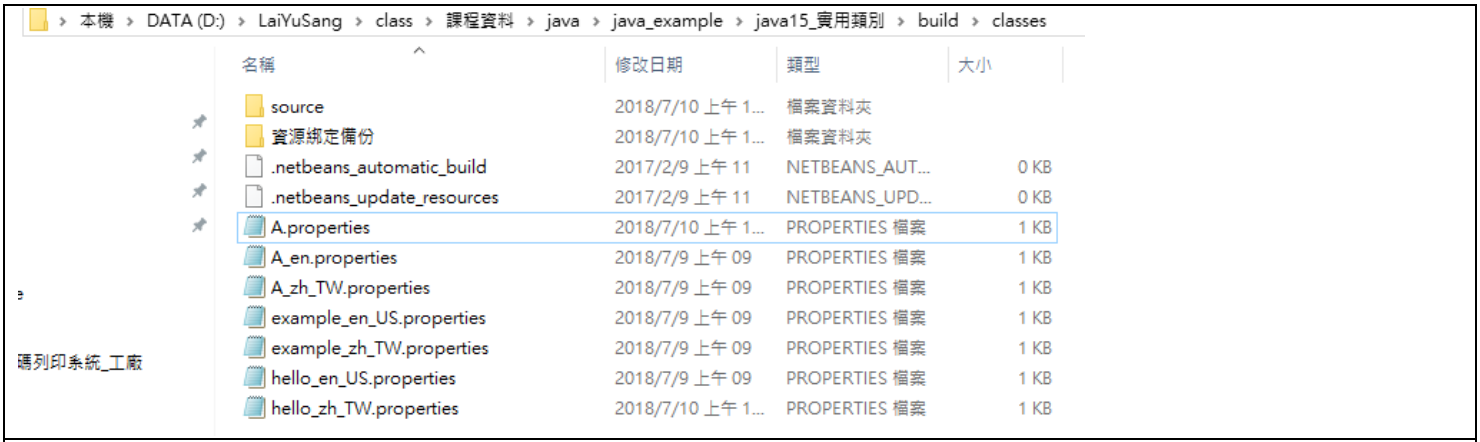
example_en_US.properties

example_zh_TW.properties

範例：

定義資源檔，放到 src 的根目錄下面

example_en_US.properties




```java
public class ResourceBundleDemo1 {
//路徑要放到 [classes] 下
    public static void main(String[] args) {
        Locale locale1 = new Locale("zh", "TW");
        ResourceBundle resb1 = ResourceBundle.getBundle("example", locale1);
        System.out.println(resb1.getString("aaa"));

        Locale locale2 = new Locale("en", "US");
        ResourceBundle resb2 = ResourceBundle.getBundle("example", locale2);
        System.out.println(resb2.getString("aaa"));
    }
}
```

example_zh_TW.properties - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

aaa = \u54C8\u56C9

example_en_US.properties - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

aaa = Hello

哈囉

Hello

```java
public class ResourceBundleDemo2 {
//路徑要放到 [classes] 下

    public static void main(String[] args) throws UnsupportedEncodingException {
        ResourceBundle resb1 = ResourceBundle.getBundle("hello", Locale.US);
        System.out.println(resb1.getString("hello"));

        ResourceBundle resb2 = ResourceBundle.getBundle("hello", Locale.TAIWAN);
        //System.out.println(rsTaiwan.getString(testString));
        System.out.println(new String(resb2.getString("hello").getBytes("8859_1"), "big5"));
    }
}
```

hello_en_US.properties - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
hello=Hello World!
```

hello_zh_TW.properties - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
hello=哈囉 世界!
```

```
Hello World!
哈囉　世界 !
```

```java
public class ResourceBundleDemo3 {
//路徑要放到 [classes] 下 , 讀不到資源檔時 , 會讀到 A.properties
    public static void main(String[] args) throws UnsupportedEncodingException {
        ResourceBundle rsdefault = ResourceBundle.getBundle("A", Locale.getDefault());
        System.out.println(new String(rsdefault.getString("Test").getBytes("8859_1"), "big5"));
        ResourceBundle rsEnglish = ResourceBundle.getBundle("A", Locale.ENGLISH);
        System.out.println(rsEnglish.getString("Test"));
        ResourceBundle rsTaiwan = ResourceBundle.getBundle("A", Locale.TAIWAN);
        //System.out.println(rsTaiwan.getString(testString));
        System.out.println(new String(rsTaiwan.getString("Test").getBytes("8859_1"), "big5"));
    }
}
```

A_zh_TW.properties - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
Test=測試它
```

A_en.properties - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
Test=Test it
```

A.properties - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
Test=xxxxxx
```

```
測試它
Test it
測試它
```

Greetings.properties - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
HELLO_MESG = Hello, everyone!
GOODBYE_MSG = Goodbye everyone!
```

```java
public class ResourceBundleDemo4 {
//路徑要放到 [classes] 下
    public static void main(String[] args) {

        ResourceBundle resource = ResourceBundle.getBundle("Greetings", Locale.US);
        System.out.println(resource.getString("HELLO_MESG"));
        System.out.println(resource.getString("GOODBYE_MSG"));
    }
}
```

```
Hello, everyone!
Goodbye everyone!
```

ISO 639: 2-letter codes

AA "Afar"
AB "Abkhazian"
AF "Afrikaans"
AM "Amharic"
AR "Arabic"
AS "Assamese"
AY "Aymara"
AZ "Azerbaijani"
BA "Bashkir"
BE "Byelorussian"
BG "Bulgarian"
BH "Bihari"
BI "Bislama"
BN "Bengali" "Bangla"
BO "Tibetan"
BR "Breton"
CA "Catalan"
CO "Corsican"
CS "Czech"
CY "Welsh"
DA "Danish"
DE "German"
DZ "Bhutani"
EL "Greek"
EN "English" "American"
EO "Esperanto"
ES "Spanish"
ET "Estonian"
EU "Basque"
FA "Persian"
FI "Finnish"
FJ "Fiji"
FO "Faeroese"
FR "French"
FY "Frisian"
GA "Irish"
GD "Gaelic" "Scots Gaelic"
GL "Galician"
GN "Guarani"
GU "Gujarati"
HA "Hausa"
HI "Hindi"
HR "Croatian"
HU "Hungarian"
HY "Armenian"
IA "Interlingua"
IE "Interlingue"
IK "Inupiak"
IN "Indonesian"
IS "Icelandic"
IT "Italian"
IW "Hebrew"
JA "Japanese"
JI "Yiddish"
JW "Javanese"
KA "Georgian"
KK "Kazakh"
KL "Greenlandic"
KM "Cambodian"
KN "Kannada"
KO "Korean"

KS "Kashmiri"
KU "Kurdish"
KY "Kirghiz"
LA "Latin"
LN "Lingala"
LO "Laothian"
LT "Lithuanian"
LV "Latvian" "Lettish"
MG "Malagasy"
MI "Maori"
MK "Macedonian"
ML "Malayalam"
MN "Mongolian"
MO "Moldavian"
MR "Marathi"
MS "Malay"
MT "Maltese"
MY "Burmese"
NA "Nauru"
NE "Nepali"
NL "Dutch"
NO "Norwegian"
OC "Occitan"
OM "Oromo" "Afan"
OR "Oriya"
PA "Punjabi"
PL "Polish"
PS "Pashto" "Pushto"
PT "Portuguese"
QU "Quechua"
RM "Rhaeto-Romance"
RN "Kirundi"
RO "Romanian"
RU "Russian"
RW "Kinyarwanda"
SA "Sanskrit"
SD "Sindhi"
SG "Sangro"
SH "Serbo-Croatian"
SI "Singhalese"
SK "Slovak"
SL "Slovenian"
SM "Samoan"
SN "Shona"
SO "Somali"
SQ "Albanian"
SR "Serbian"
SS "Siswati"
ST "Sesotho"
SU "Sudanese"
SV "Swedish"
SW "Swahili"
TA "Tamil"
TE "Tegulu"
TG "Tajik"
TH "Thai"
TI "Tigrinya"
TK "Turkmen"
TL "Tagalog"
TN "Setswana"
TO "Tonga"
TR "Turkish"
TS "Tsonga"

TT "Tatar"
TW "Twi"
UK "Ukrainian"
UR "Urdu"
UZ "Uzbek"
VI "Vietnamese"
VO "Volapuk"
WO "Wolof"
XH "Xhosa"
YO "Yoruba"
ZH "Chinese"
ZU "Zulu"

Java Locale List

Sr.	Locale	Language	Country
1	ms_MY	Malay (Malaysia)	Malaysia
2	ar_QA	Arabic (Qatar)	Qatar
3	is_IS	Icelandic (Iceland)	Iceland
4	fi_FI	Finnish (Finland)	Finland
5	pl	Polish	
6	en_MT	English (Malta)	Malta
7	it_CH	Italian (Switzerland)	Switzerland
8	nl_BE	Dutch (Belgium)	Belgium
9	ar_SA	Arabic (Saudi Arabia)	Saudi Arabia
10	ar_IQ	Arabic (Iraq)	Iraq
11	es_PR	Spanish (Puerto Rico)	Puerto Rico
12	es_CL	Spanish (Chile)	Chile
13	fi	Finnish	
14	de_AT	German (Austria)	Austria
15	da	Danish	
16	en_GB	English (United Kingdom)	United Kingdom
17	es_PA	Spanish (Panama)	Panama
18	sr	Serbian	
19	ar_YE	Arabic (Yemen)	Yemen
20	mk_MK	Macedonian (Macedonia)	Macedonia
21	mk	Macedonian	

22	en_CA	English (Canada)	Canada
23	vi_VN	Vietnamese (Vietnam)	Vietnam
24	nl_NL	Dutch (Netherlands)	Netherlands
25	es_US	Spanish (United States)	United States
26	zh_CN	Chinese (China)	China
27	es_HN	Spanish (Honduras)	Honduras
28	en_US	English (United States)	United States
29	fr	French	
30	th	Thai	
31	ar	Arabic	
32	ar_MA	Arabic (Morocco)	Morocco
33	lv	Latvian	
34	de	German	
35	in_ID	Indonesian (Indonesia)	Indonesia
36	hr	Croatian	
37	en_ZA	English (South Africa)	South Africa
38	ko_KR	Korean (South Korea)	South Korea
39	ar_TN	Arabic (Tunisia)	Tunisia
40	in	Indonesian	
41	ja	Japanese	
42	sr_RS	Serbian (Serbia)	Serbia
43	be_BY	Belarusian (Belarus)	Belarus
44	zh_TW	Chinese (Taiwan)	Taiwan
45	ar_SD	Arabic (Sudan)	Sudan
46	pt	Portuguese	
47	is	Icelandic	
48	ja_JP_JP_#u-ca-japanese	Japanese (Japan，JP)	Japan
49	es_BO	Spanish (Bolivia)	Bolivia
50	ar_DZ	Arabic (Algeria)	Algeria
51	ms	Malay	

52	es_AR	Spanish (Argentina)	Argentina
53	ar_AE	Arabic (United Arab Emirates)	United Arab Emirates
54	fr_CA	French (Canada)	Canada
55	sl	Slovenian	
56	es	Spanish	
57	lt_LT	Lithuanian (Lithuania)	Lithuania
58	sr_ME_#Latn	Serbian (Latin，Montenegro)	Montenegro
59	ar_SY	Arabic (Syria)	Syria
60	ru_RU	Russian (Russia)	Russia
61	fr_BE	French (Belgium)	Belgium
62	es_ES	Spanish (Spain)	Spain
63	bg	Bulgarian	
64	iw_IL	Hebrew (Israel)	Israel
65	sv	Swedish	
66	en	English	
67	iw	Hebrew	
68	da_DK	Danish (Denmark)	Denmark
69	es_CR	Spanish (Costa Rica)	Costa Rica
70	zh_HK	Chinese (Hong Kong)	Hong Kong
71	zh	Chinese	
72	ca_ES	Catalan (Spain)	Spain
73	th_TH	Thai (Thailand)	Thailand
74	uk_UA	Ukrainian (Ukraine)	Ukraine
75	es_DO	Spanish (Dominican Republic)	Dominican Republic
76	es_VE	Spanish (Venezuela)	Venezuela
77	pl_PL	Polish (Poland)	Poland
78	ar_LY	Arabic (Libya)	Libya
79	ar_JO	Arabic (Jordan)	Jordan
80	it	Italian	
81	uk	Ukrainian	

82	hu_HU	Hungarian (Hungary)	Hungary
83	ga	Irish	
84	es_GT	Spanish (Guatemala)	Guatemala
85	es_PY	Spanish (Paraguay)	Paraguay
86	bg_BG	Bulgarian (Bulgaria)	Bulgaria
87	hr_HR	Croatian (Croatia)	Croatia
88	sr_BA_#Latn	Serbian (Latin，Bosnia and Herzegovina)	Bosnia and Herzegovina
89	ro_RO	Romanian (Romania)	Romania
90	fr_LU	French (Luxembourg)	Luxembourg
91	no	Norwegian	
92	lt	Lithuanian	
93	en_SG	English (Singapore)	Singapore
94	es_EC	Spanish (Ecuador)	Ecuador
95	sr_BA	Serbian (Bosnia and Herzegovina)	Bosnia and Herzegovina
96	es_NI	Spanish (Nicaragua)	Nicaragua
97	sk	Slovak	
98	ru	Russian	
99	mt	Maltese	
100	es_SV	Spanish (El Salvador)	El Salvador
101	nl	Dutch	
102	hi_IN	Hindi (India)	India
103	et	Estonian	
104	el_GR	Greek (Greece)	Greece
105	sl_SI	Slovenian (Slovenia)	Slovenia
106	it_IT	Italian (Italy)	Italy
107	ja_JP	Japanese (Japan)	Japan
108	de_LU	German (Luxembourg)	Luxembourg
109	fr_CH	French (Switzerland)	Switzerland
110	mt_MT	Maltese (Malta)	Malta
111	ar_BH	Arabic (Bahrain)	Bahrain

112	sq	Albanian	
113	vi	Vietnamese	
114	sr_ME	Serbian (Montenegro)	Montenegro
115	pt_BR	Portuguese (Brazil)	Brazil
116	no_NO	Norwegian (Norway)	Norway
117	el	Greek	
118	de_CH	German (Switzerland)	Switzerland
119	zh_SG	Chinese (Singapore)	Singapore
120	ar_KW	Arabic (Kuwait)	Kuwait
121	ar_EG	Arabic (Egypt)	Egypt
122	ga_IE	Irish (Ireland)	Ireland
123	es_PE	Spanish (Peru)	Peru
124	cs_CZ	Czech (Czech Republic)	Czech Republic
125	tr_TR	Turkish (Turkey)	Turkey
126	cs	Czech	
127	es_UY	Spanish (Uruguay)	Uruguay
128	en_IE	English (Ireland)	Ireland
129	en_IN	English (India)	India
130	ar_OM	Arabic (Oman)	Oman
131	sr_CS	Serbian (Serbia and Montenegro)	Serbia and Montenegro
132	ca	Catalan	
133	be	Belarusian	
134	sr__#Latn	Serbian (Latin)	
135	ko	Korean	
136	sq_AL	Albanian (Albania)	Albania
137	pt_PT	Portuguese (Portugal)	Portugal
138	lv_LV	Latvian (Latvia)	Latvia
139	sr_RS_#Latn	Serbian (Latin，Serbia)	Serbia
140	sk_SK	Slovak (Slovakia)	Slovakia
141	es_MX	Spanish (Mexico)	Mexico

142	en_AU	English (Australia)	Australia
143	no_NO_NY	Norwegian (Norway，Nynorsk)	Norway
144	en_NZ	English (New Zealand)	New Zealand
145	sv_SE	Swedish (Sweden)	Sweden
146	ro	Romanian	
147	ar_LB	Arabic (Lebanon)	Lebanon
148	de_DE	German (Germany)	Germany
149	th_TH_TH_#u-nu-thai	Thai (Thailand，TH)	Thailand
150	tr	Turkish	
151	es_CO	Spanish (Colombia)	Colombia
152	en_PH	English (Philippines)	Philippines
153	et_EE	Estonian (Estonia)	Estonia
154	el_CY	Greek (Cyprus)	Cyprus
155	hu	Hungarian	
156	fr_FR	French (France)	France

- ，

ar_AE - 阿拉伯文 (阿拉伯聯合大公國)， 阿拉伯聯合大公國

ar_JO - 阿拉伯文 (約旦)， 約旦

ar_SY - 阿拉伯文 (敘利亞)， 敘利亞

hr_HR - 克羅埃西亞文 (克羅埃西亞)， 克羅埃西亞

fr_BE - 法文 (比利時)， 比利時

es_PA - 西班牙文 (巴拿馬)， 巴拿馬

mt_MT - 馬爾他文 (馬爾他)， 馬爾他

es_VE - 西班牙文 (委內瑞拉)， 委內瑞拉

bg - 保加利亞文，

zh_TW - 中文 (台灣)， 台灣

it - 義大利文，

ko - 韓文，

uk - 烏克蘭文，

lv - 拉脫維亞文 (列特文)，

da_DK - 丹麥文 (丹麥)， 丹麥

es_PR - 西班牙文 (波多黎各)， 波多黎各

vi_VN - 越南文 (越南)， 越南

en_US - 英文 (美國)， 美國

sr_ME - 塞爾維亞文 (蒙特尼格羅)， 蒙特尼格羅

sv_SE - 瑞典文 (瑞典)， 瑞典

es_BO - 西班牙文 (玻利維亞)， 玻利維亞

en_SG - 英文 (新加坡)， 新加坡

ar_BH - 阿拉伯文 (巴林)， 巴林

pt - 葡萄牙文，

ar_SA - 阿拉伯文 (沙烏地阿拉伯)， 沙烏地阿拉伯

sk - 斯洛伐克文，

ar_YE - 阿拉伯文 (葉門)， 葉門

hi_IN - 北印度文 (印度)， 印度

ga - 愛爾蘭文，

en_MT - 英文 (馬爾他)， 馬爾他

fi_FI - 芬蘭文 (芬蘭)， 芬蘭

et - 愛沙尼亞文，

sv - 瑞典文，

cs - 捷克文，

sr_BA_#Latn - 塞爾維亞文 (拉丁文，波士尼亞赫塞哥維納)， 波士尼亞赫塞哥維納

el - 希臘文，

uk_UA - 烏克蘭文 (烏克蘭)， 烏克蘭

hu - 匈牙利文，

fr_CH - 法文 (瑞士)， 瑞士

in - 印尼文，

es_AR - 西班牙文 (阿根廷)， 阿根廷

ar_EG - 阿拉伯文 (埃及)， 埃及

ja_JP_JP_#u-ca-japanese - 日文 (日本，JP)， 日本

es_SV - 西班牙文 (薩爾瓦多)， 薩爾瓦多

pt_BR - 葡萄牙文 (巴西)， 巴西

be - 白俄羅斯文，

is_IS - 冰島文 (冰島)， 冰島

cs_CZ - 捷克文 (捷克共和國)， 捷克共和國

es - 西班牙文，

pl_PL - 波蘭文 (波蘭)， 波蘭

tr - 土耳其文，

ca_ES - 嘉泰羅尼亞文 (西班牙)， 西班牙

sr_CS - 塞爾維亞文 (塞爾維亞及蒙特尼哥羅)， 塞爾維亞及蒙特尼哥羅

ms_MY - 馬來文 (馬來西亞)， 馬來西亞

hr - 克羅埃西亞文，

lt - 立陶宛文，

es_ES - 西班牙文 (西班牙)， 西班牙

es_CO - 西班牙文 (哥倫比亞)， 哥倫比亞

bg_BG - 保加利亞文 (保加利亞)， 保加利亞

sq - 阿爾巴尼亞文，

fr - 法文，

ja - 日文，

sr_BA - 塞爾維亞文 (波士尼亞赫塞哥維納)， 波士尼亞赫塞哥維納

is - 冰島文，
es_PY - 西班牙文 (巴拉圭)， 巴拉圭
de - 德文，
es_EC - 西班牙文 (厄瓜多爾)， 厄瓜多爾
es_US - 西班牙文 (美國)， 美國
ar_SD - 阿拉伯文 (蘇丹)， 蘇丹
en - 英文，
ro_RO - 羅馬尼亞文 (羅馬尼亞)， 羅馬尼亞
en_PH - 英文 (菲律賓)， 菲律賓
ca - 嘉泰羅尼亞文，
ar_TN - 阿拉伯文 (突尼西亞)， 突尼西亞
sr_ME_#Latn - 塞爾維亞文 (拉丁文，蒙特尼格羅)， 蒙特尼格羅
es_GT - 西班牙文 (瓜地馬拉)， 瓜地馬拉
sl - 斯拉維尼亞文，
ko_KR - 韓文 (南韓)， 南韓
el_CY - 希臘文 (賽普勒斯)， 賽普勒斯
es_MX - 西班牙文 (墨西哥)， 墨西哥
ru_RU - 俄文 (俄羅斯聯邦)， 俄羅斯聯邦
es_HN - 西班牙文 (宏都拉斯)， 宏都拉斯
zh_HK - 中文 (香港)， 香港
no_NO_NY - 挪威文 (挪威，Nynorsk)， 挪威
hu_HU - 匈牙利文 (匈牙利)， 匈牙利
th_TH - 泰文 (泰國)， 泰國
ar_IQ - 阿拉伯文 (伊拉克)， 伊拉克
es_CL - 西班牙文 (智利)， 智利
fi - 芬蘭文，
ar_MA - 阿拉伯文 (摩洛哥)， 摩洛哥
ga_IE - 愛爾蘭文 (愛爾蘭)， 愛爾蘭
mk - 馬其頓文，
tr_TR - 土耳其文 (土耳其)， 土耳其
et_EE - 愛沙尼亞文 (愛沙尼亞)， 愛沙尼亞
ar_QA - 阿拉伯文 (卡達)， 卡達
sr__#Latn - 塞爾維亞文 (拉丁文)，
pt_PT - 葡萄牙文 (葡萄牙)， 葡萄牙
fr_LU - 法文 (盧森堡)， 盧森堡
ar_OM - 阿拉伯文 (阿曼)， 阿曼
th - 泰文，
sq_AL - 阿爾巴尼亞文 (阿爾巴尼亞)， 阿爾巴尼亞
es_DO - 西班牙文 (多明尼加)， 多明尼加
es_CU - 西班牙文 (古巴)， 古巴
ar - 阿拉伯文，
ru - 俄文，
en_NZ - 英文 (紐西蘭)， 紐西蘭

sr_RS - 塞爾維亞文 (塞爾維亞)， 塞爾維亞

de_CH - 德文 (瑞士)， 瑞士

es_UY - 西班牙文 (烏拉圭)， 烏拉圭

ms - 馬來文，

el_GR - 希臘文 (希臘)， 希臘

iw_IL - 希伯來文 (以色列)， 以色列

en_ZA - 英文 (南非)， 南非

th_TH_TH_#u-nu-thai - 泰文 (泰國，TH)， 泰國

hi - 北印度文，

fr_FR - 法文 (法國)， 法國

de_AT - 德文 (奧地利)， 奧地利

nl - 荷蘭文，

no_NO - 挪威文 (挪威)， 挪威

en_AU - 英文 (澳大利亞)， 澳大利亞

vi - 越南文，

nl_NL - 荷蘭文 (荷蘭)， 荷蘭

fr_CA - 法文 (加拿大)， 加拿大

lv_LV - 拉脫維亞文 (列特文) (拉脫維亞)， 拉脫維亞

de_LU - 德文 (盧森堡)， 盧森堡

es_CR - 西班牙文 (哥斯大黎加)， 哥斯大黎加

ar_KW - 阿拉伯文 (科威特)， 科威特

sr - 塞爾維亞文，

ar_LY - 阿拉伯文 (利比亞)， 利比亞

mt - 馬爾他文，

it_CH - 義大利文 (瑞士)， 瑞士

da - 丹麥文，

de_DE - 德文 (德國)， 德國

ar_DZ - 阿拉伯文 (阿爾及利)， 阿爾及利

sk_SK - 斯洛伐克文 (斯洛伐克)， 斯洛伐克

lt_LT - 立陶宛文 (立陶宛)， 立陶宛

it_IT - 義大利文 (義大利)， 義大利

en_IE - 英文 (愛爾蘭)， 愛爾蘭

zh_SG - 中文 (新加坡)， 新加坡

ro - 羅馬尼亞文，

en_CA - 英文 (加拿大)， 加拿大

nl_BE - 荷蘭文 (比利時)， 比利時

no - 挪威文，

pl - 波蘭文，

zh_CN - 中文 (中國)， 中國

ja_JP - 日文 (日本)， 日本

de_GR - 德文 (希臘)， 希臘

sr_RS_#Latn - 塞爾維亞文 (拉丁文，塞爾維亞)， 塞爾維亞

iw - 希伯來文，

en_IN - 英文 (印度)， 印度

ar_LB - 阿拉伯文 (黎巴嫩)， 黎巴嫩

es_NI - 西班牙文 (尼加拉瓜)， 尼加拉瓜

zh - 中文，

mk_MK - 馬其頓文 (馬其頓)， 馬其頓

be_BY - 白俄羅斯文 (白俄羅斯)， 白俄羅斯

sl_SI - 斯拉維尼亞文 (斯洛維尼亞)， 斯洛維尼亞

es_PE - 西班牙文 (祕魯共和國)， 祕魯共和國

in_ID - 印尼文 (印尼)， 印尼

en_GB - 英文 (英國)， 英國

字母	日期或时间元素	表示	示例
G	Era 标志符	Text	AD
y	年	Year	1996; 96
M	年中的月份	Month	July; Jul; 07
w	年中的周数	Number	27
W	月份中的周数	Number	2
D	年中的天数	Number	189
d	月份中的天数	Number	10
F	月份中的星期	Number	2
E	星期中的天数	Text	Tuesday; Tue
a	Am/pm 标记	Text	PM
H	一天中的小时数（0-23）	Number	0
k	一天中的小时数（1-24）	Number	24
K	am/pm 中的小时数（0-11）	Number	0
h	am/pm 中的小时数（1-12）	Number	12
m	小时中的分钟数	Number	30
s	分钟中的秒数	Number	55
S	毫秒数	Number	978
z	时区	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	时区	RFC 822 time zone	-0800