第8章　常用类

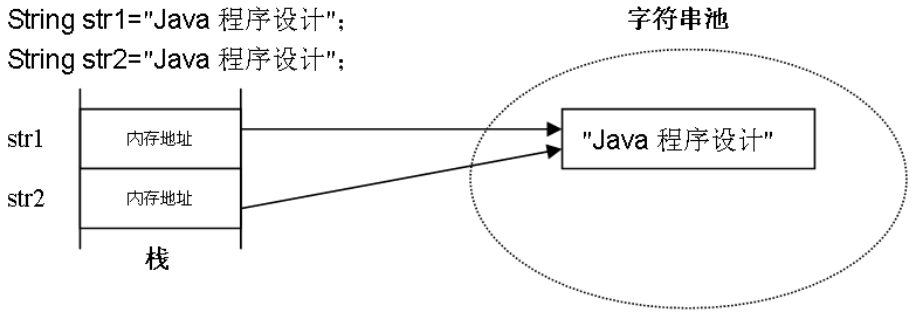
- 8.1 String类
- 8.2 StringBuffer类
- 8.3 Math类
- 8.4 包装类
- 8.5 日期日历类

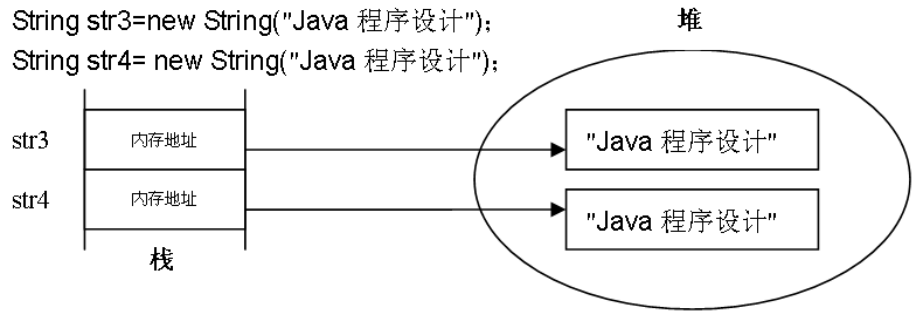
8.1 String类

8.1.1字符串的概念

1．字符串**常量**：即是用双引号(" ")括住的字符序列。这种表示法简单、实用，例如：

String str1="Java程序设计"；//str1是String类的实例

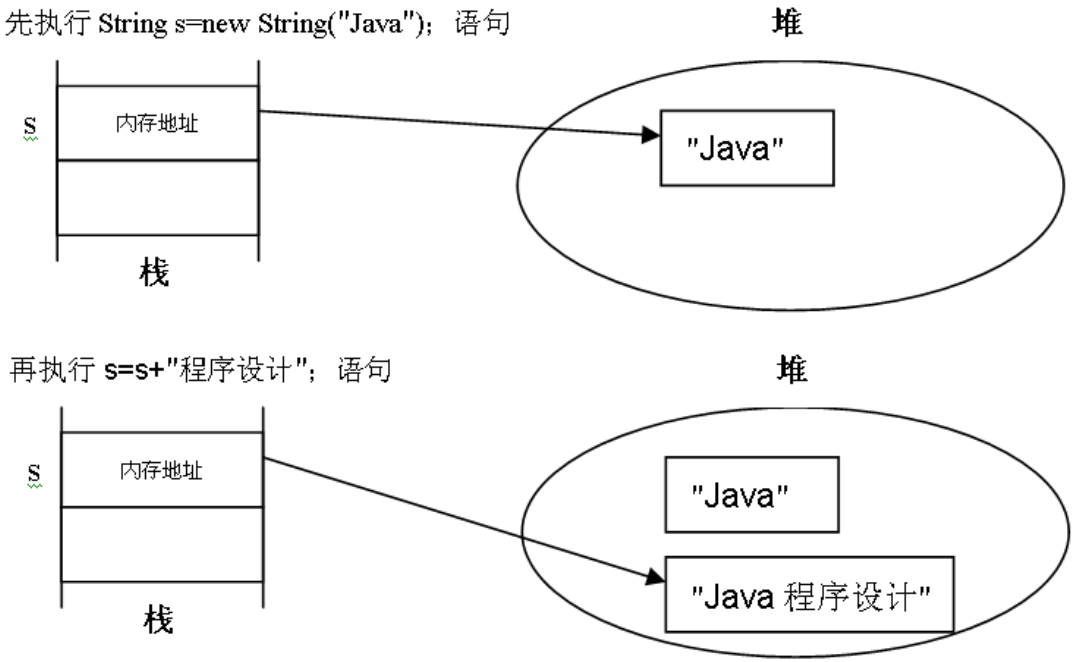




8.1 String类

8.1.1字符串的概念

2．**字符串的内容不可改变**。这就是说，字符串一旦生成，它的值及所分配的内存空间就不能再被改变。如果硬性改变其值，将会产生一个新的字符串，**原对象引用所指的内容不会随之变化**。



字符串

- Java中的两类字符串：
  - String类
    - 实例不能修改，即常量字符串
  - StringBuffer类
    - 又称缓冲字符串，实例可修改串内容

8.1.2字符串的创建

1．String()：默认构造方法，生成一个空串

2．String(String original)：以一个字符串为参数构造另一个字符串，即进行字符串拷贝

3．以字符数组为参数构造字符串

(1) String(char[] value)　　其中：value为源字符数组

(2) String(char[] value,int offset,int count) 其中：value含义同(1)，offset是value的开始下标，count是字符个数

8.以字节数组为参数构造字符串：

(1) String(byte[] bytes)

(2) String(byte[] bytes, String charsetName)

(3) String(byte[] bytes, int offset, int length)

(4) String(byte[] bytes, int offset, int length, String charsetName)

5．String(StringBuffer buffer)：用缓冲字符串为参数构造字符串

8.1 String类

8.1.3字符串的常用方法

1.获取字符串的长度

8.字符串的查找

2.字符串的比较

5.字符串的修改

3.字符串的提取

6.字符串与基本类型的互换

字符串

- 获取字符串的相关信息：
  - 获取字符串的当前长度

    String str=“This is a String”;

    int len=str.length();

    （StringBuffer的空间可变，其中capacity()方法与length()类似，测试StringBuffer的内存空间的大小）

字符串

- 获取字符串的相关信息：
  - 在字符串中进行搜索：
    - indexOf　若返回-1表示不存在搜寻字符
    - lastIndexOf
    - 例如：

```
String str="This is a String";
int index1=str.indexOf("is");       //index1==2
int index2=str.indexOf("is",index1+1);
int index3=str.lastIndexOf("is");//index3=?
```

字符串

- String对象的比较
  - equals();

  例： String str="This is a String";

  boolean rs=str.equals("This is another string");

    - 注意：String对象中的==与equals方法的区别：
      - ==表示两个对象是否是同一对象体的引用；
      - 覆盖Object的equals方法：引用的对象内容一致，即相等

字符串

- 提取字符串中部分内容
  - getChars(int srcBegin,int srcEnd,char[] dst,int dtsBegin);

    例如：String str=“This is a string”;

    char[] chr=new char[10];

    str.getChars(5,12,chr,0);
  - charAt(int);

    例如：String str=“This is a String”;

    char chr=str.charAt(3);
  - substring(int beg,int end); //不包括end

    String str= “This is a String”.substring(2,4);

字符串

- 字符串与字符数组-字符串转成字符数组 toCharArray();

  如：String str=“hello”;

  char[] chs=Str.toCharArray();

  ---》[‘h’,’e’,’l’,’l’,’o’]

  -字符数组转成字符串

  char[] chrs={'h','e','l'};

  String h3=new String(chrs);

注意：java中字符串与字符数组不能直接相互赋值 StringTestExample.java

思考一下前面的substring可以如何实现？

字符串

- 字符串操作：

  -split:字符串进行分割多个子串，返回的是字符串数组

```
String str="d:mydocument:file:word.doc";
String[] strs=str.split(":");
```

特殊：

```
String str="d:\\mydocument\\file\\word.doc";
String[] strs=str.split("\\\\");
```

综合示例

- “this is an economical report”
  - 有多少个字符？有多少个英文字符？
  - 有多少种字符？有多少种英文字符？
  - 每种字符的个数？
  - 有多少个词？

见StringTestExample类

字符串

- 字符串其它常用操作：

-toUpperCase()和toLowerCase()：大小写字母的切换

– concat：将两个字符串合并成一个字符串

/*也可以直接用＋运算符进行*/

– replace：将字符串中的某个字符串替换成另一个字符串

- 例如：

String str1=“This is a String”;

String str2=str1.replace(‘T’,’t’);

- 字符串操作：
  - 空格处理 trim()
  - 去除的是头尾空格
  - 若要去除字符串内部空格呢？

见StringTestExample类

字符串

- 字符串操作：
  - String提供了静态方法：valueOf可以将任意类型的对象转换成一个字符串
    - 例如：System.out.println(String.valueOf(Math.PI));

示例

- 编写一个程序完成下列功能：
  - 创建一个String类的对象str，其内容为“you Are a GOOD boy”；
  - 在屏幕上显示str的内容；
  - 将字符串的首字符改为大写，其余均变成小写，并将修改后的内容显示在屏幕上；
  - 在屏幕上显示字符串str的长度、第四个字符；
  - 在屏幕上显示字符串str的最后一个单词boy

字符串

- 字符串操作举例：——用串进行
  - 已知以分号；间隔若干姓名
  - 将输入姓名中包含“Wang”的替换成“Zhang”
  - 将输入姓名按倒序输出
- 判断用户输入的Email地址是否合法
  - 邮箱地址肯定有@字符
  - 如果在后三个字符中出现@字符，输入也不合法
  - 如果邮箱地址中@的个数不止一个，也不合法
  - 如果@字符后没有“.”字符，或“.”后只有一个字符，则邮箱地址不合法

8.2 StringBuffer类

8.2.1 引入StringBuffer的原因

StringBuffer也是一个字符序列，类似于String，但与String不同的是：

◆可以改变其长度和内容，用户可以根据需要，在StringBuffer中进行附加、插入、替换、删除、查询等操作；

◆操作结果作用于StringBuffer串本身，并无新对象产生，非常适合大型文本的处理。

8.2 StringBuffer类

8.2.2 StringBuffer对象的创建

Java提供了三个常用构造方法来创建StringBuffer对象，具体如下：

1.StringBuffer( )：建立一个不包含任何文本的StringBuffer对象，可在以后操作时添加其内容。初始容量为16字节

2.StringBuffer(int capacity)：建立一个容量为capacity的StringBuffer对象，它不包含任何文本

3.StringBuffer (String str)：以参数str来创建StringBuffer对象

说明：随着文本的增加，字符串的长度在不断增大；当长度大于StringBuffer对象的现有容量时，Java会自动增加其容量。所以，在进行StringBuffer的增加、删除操作时，不必考虑其容量问题。

8.2 StringBuffer类

8.2.2 StringBuffer对象的创建

比较String和StringBuffer在修改字符串中的效率

参见StringBufferTest类

8.2 StringBuffer类

8.2.3 StringBuffer类的常用方法

1.对象自身操作

2.增加字符串

3.删除字符或字符串

8.替换字符串

5.获取或设置指定位置的字符

6.获取字符串的子串

7.将包含的字符串逆序

8.将StringBuffer对象转换对象

8.3 Math类

8.3.1 Math类的功能

Math类位于java.lang包中，它继承了Object类，包含基本的数学计算，如指数、对数、平方根和三角函数，由于它是final类，不能再被继承。Math类的属性、方法绝大多数是静态(static)的，在使用时不必创建对象，直接采用：Math.属性 或 Math.方法([参数表]) 格式调用即可。

1.静态常量

(1) E：e 的近似值，为double类型。

(2) PI：圆周率 的近似值，为double类型。

8.3 Math 类(选学)

8.3.1 Math 类的功能

2.常用方法：

(1)求绝对值：返回值类型 abs(参数)

(2)求两个数中的较大者：返回值类型 max(参数1, 参数2)

(3)求两个数中的较小者：返回值类型 min(参数1, 参数2) 说明同(2)

(4)将实数四舍五入为整数：

(5)求平方根：double sqrt(double a) 当a小于0或NaN时，返回NaN

(6)求：double exp(double x)

(7)求：double pow(double x,double y)

(8)求对数

(9)生成[0，1)的随机小数：double random()，利用它进行适当变换后可得任意区间的随机整数

(10) 求三角函数与反三角函数值：

8.3 Math 类(选学)

8.3.2 Math类的应用举例

在Math类的众多方法中，随机数生成方法random()的使用比较灵活，利用它可以模拟随机事件的发生，例如：摸奖、发扑克牌等。

random()只能生成[0，1)的随机小数，若要生成指定区间的随机整数，需要进行放大、平移、取整等操作，具体如下：

设a、b分别为两个整数(且a<=b)，由于Math.random()值在[0, 1)，那么，(b-a+1)*Math.random()的值会在[0, b-a+1)范围内；加上a进行平移操作，a+(b-a+1)*Math.random()的值将在[a, b+1)中；最后取整，得到[a,b]范围的随机整数。

示例

- 编程完成下列功能：
  - 要求生成一个随机整数x，其取值范围为1800~2004之间，表示一个年份；
  - 判断x是否是闰年，如果是，则向屏幕显示闰年的信息，否则显示不是闰年的信息

8.4 包装类

8.8.1 为什么要引入包装类

包装类其实是简称，严格意义上说，应该是基本数据类型的包装类，它们为基本数据类型提供类的功能，共包含有8个类，即Boolean、Byte、Short、Integer、Long 、Character、Float、Double，分别对应着8种基本数据类型。你是否已经看出这些类的命名规律？除了Integer、Character外，其余6个类的类名都是将对应的基本数据类型名的首字母大写后得到。

包装类位于java.lang包中，不需要使用import语句来导入。由于包装类的成员个数较多，如果逐一讲解，会导致篇幅大、重复内容多的后果，因此，我们采用“先同后异”的方式来介绍，即先讲解包装类的共同特征，再指出个别类的特殊之处。

8.4 包装类

8.8.2 包装类的类常量、构造器和常用方法

类常量

Bealoon类用TRUE、FALSE两个常量来表示“真”、“假”，其它7个类分别用MINX_VALUE、MAX_VALUE来表示对应基本类型的最小值、最大值。

构造方法

(1) 所有的包装类都可以用其对应的基本类型数据为参数，来构造相应对象。

(2) 除Character外，都提供了以String类型数据为参数的构造方法。

常用方法

(1) 将包装类对象转换为基本类型数据。

(2) 包装类对象与字符串的相互转换。

(3) 基本类型数据与字符串的相互转换。

Java类型包装类

- 类型包装类：
  - 常用方法：
    - 基本类型与串：
      - boolean getBoolean(String s);
      - byte parseByte(String s);
      - double parseDouble(String s);
      - short parseShort(String s);
      - int parseInt(String s);
      - float parseFloat(String s);
      - long parseLong(String s);

8.5.3 Character类的独特之处

现选择其中几个来示范，请予以注意：

(1)static boolean isDigit(char ch)：判断ch是否为数字。

(2)static boolean isLetter(char ch)：判断ch是否为字母。

(3)static boolean isLetterOrDigit(char ch)：判断ch是否为字母或数字。

(4)static boolean isLowerCase(char ch)：判断ch是否为小写字符。

(5)static boolean isUpperCase(char ch)：判断ch是否为大写字符。

(6)static boolean isSpaceChar(char ch)：判断ch是否为空白字符。

(7)static char toLowerCase(char ch)：将ch转换为小写字符。

(8)static char toUpperCase(char ch)：将ch转换为大写字符。

示例

- 编程完成下列功能：
  - 从命令行接收两个数据，将这两个数字相加，并在屏幕上显示相加后的结果
  - 利用输入语句从键盘上输入一个字符（＋-*/），根据输入的运算符分别进行不同的运算。

8.5 日期日历类

8.5.1 Date类

Date类位于java.util包中，它代表的是时间轴上的一个点，用一个long型的数据来度量，该数据是Date对象代表的时点距离GMT(格林尼治标准时间)1970年1月1日00时00分00秒的毫秒数。Date类具有操作时间的基本功能，例如：获取系统当前时间。由于该类在设计上存在严重缺陷，它的多个方法已过时、废弃，相关功能已转移到其它类中实现。因此，我们只介绍它的基本用法。

8.5 日期日历类

8.5.1 Date类

1．构造方法：

(1) Date()：构造日期对象，代表的是系统当前时间

(2) Date(long date)：用long型参数构造对象。参数date是指距离GMT 1970年1月1日00时00分00秒时点的长度，单位为毫秒。

2．常用方法：

(1) boolean after(Date when)：判断当前对象代表的时点是否晚于when代表的时点。

(2) boolean before(Date when)：判断当前对象代表的时点是否早于when代表的时点。

(3) long getTime()：返回当前对象代表的时点距离GMT 1970年1月1日00时00分00秒时点的毫秒数。

(4) void setTime(long time)：用参数重新设置时点，新时点距离GMT 1970年1月1日00时00分00秒时点的长度为time毫秒。

8.5 日期日历类

8.5.2 Calendar类

Date类由于设计上的缺陷，除了表示系统当前时间和日期格式化外，其它场合的应用并不多。要获取、设置日期时间，更多用到的是Calendar及其子类。

- Calendar类位于java.util包中，它具有比Date类更强大的功能。
- Calendar是抽象类，且Calendar类的构造方法是protected的,不能直接用new关键字来创建对象，但它提供了一个静态工厂方法getInstance()来得到其子类对象，例如：

```
Calendar  rightNow = Calendar.getInstance();
          //默认是当前时间
```

8.5 日期日历类

8.5.2 Calendar类

Calendar类创建指定日期：

```
Calendar  rightNow = Calendar.getInstance();
rightNow.set(2014,10-1,12);//月从0开始
```

public final void set(int year,int month,int date)

8.5 日期日历类

8.5.2 Calendar类

1.Calendar类常量

(1) 星期几：SUNDAY、 MONDAY、 TUESDAY、 WEDNESDAY、 THURSDAY、 FRIDAY、 SATURDAY。

(2) 月份：JANUARY、 FEBRUARY、 MARCH、 APRIL、 MAY、 JUNE、 JULY、 AUGUST、 SEPTEMBER、 OCTOBER、 NOVEMBER、 DECEMBER。

(3) 上午、下午、上午_下午：AM、PM、AM_PM。

(4) 年、月、日、时、分、秒：YEAR、MONTH、DATE、HOUR、MINUTE、SECOND。

(5) 一天中的第几个小时(0-23)：HOUR_OF_DAY 。

8.5 日期日历类

8.5.2 Calendar类

1.Calendar常用方法

public void set(int field,int value)
public void get(int field)
public final Date getTime() //与Date类不同，它返回的是Date类型.

8.5.2 Calendar类

1.Calendar常用方法

```
Calendar c1 = Calendar.getInstance();
c1.set(Calendar.DATE,10);
c1.set(Calendar.YEAR,2008);
//
把c1对象的日期加上10，也就是c1所表的日期的10天后的日期，其它所有的数值会被重新计算
c1.add(Calendar.DATE, 10);
c1.add(Calendar.DATE, -10);

int year = c1.get(Calendar.YEAR);  // 获得年份
int month = c1.get(Calendar.MONTH) + 1;  // 获得月份
int date = c1.get(Calendar.DATE); // 获得日期
//
获得星期几：1代表星期日、2代表星期1，以此类推（与Date类不同）
int day = c1.get(Calendar.DAY_OF_WEEK);
```

GregorianCalendar类选学

8.5 日期日历类

8.5.2 GregorianCalendar类

Calendar的子类GregorianCalendar类在编程中经常使用，它提供了操作日期、时间更具体、更高效的方法。

Gregorian历就是我们现在使用的公历，由罗马教皇格里高利十三世正式颁布，史称格里高利历。

GregorianCalendar类是Calendar的一个具体子类，位于java.util包中，它支持多种日历，功能强大。

8.5 日期日历类

8.5.2 GregorianCalendar类

2.GregorianCalendar类构造方法

(1) GregorianCalendar()：使用系统当前时间来构造对象

(2) GregorianCalendar(int year, int month, int dayOfMonth)：使用年、月、日来构造对象

(3) GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute)：使用年、月、日、时、分来构造对象

```
GregorianCalendar gc = new GregorianCalendar();

GregorianCalendar gc = new GregorianCalendar(2009,6-1,12);
```

8.5 日期日历类

8.5.2 GregorianCalendar类

3.GregorianCalendar类常用方法

(1) int get (int field)：得到指定字段的值，参数field通常用常量表示。

(2) void set(int field, int val)：设置指定字段的值，参数要求同上。

(3) Date getTime()：得到对应的Date对象。

(4)long getTimeInMillis():返回距离GMT1970.1.1 00:00:00的毫秒数。

8.5 日期日历类

8.6.3 SimpleDateFormat类

SimpleDateFormat类是DateFormat的一个具体子类，位于java.text包中。该类具有两大转换功能，一是按用户设置的格式来输出日期，实现从日期到文本的转换；二是将文本解析为日期，实现从文本到日期的转换。用户通过设置“输出模式”可控制输出日期的格式，输出模式为字符串形式，由模式字母和固定字符组成。

表 4-1　　常用模式字母及其含义

模式字母	含义	模式字母	含义	模式字母	含义	模式字母	含义
y	年份	M	月份	d	天数	E	星期
h	小时	m	分钟	s	秒钟	a	Am/Pm

SimpleDateFormat常用，需要熟练使用

8.5 日期日历类

8.6.3 SimpleDateFormat类

1．构造方法

(1) SimpleDateFormat()：使用系统默认的模式来构造对象。

(2) SimpleDateFormat(String pattern)：使用设定的模式来构造对象。

```
SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-
dd hh:mm:ss");
```

2．常用方法

(1) void applyPattern(String pattern)：设置输出模式。

(2) String format(Date date)：将日期按指定模式输出，结果为字符串类型。

(3) Date parse(String source)：将日期形式的字符串转换成Date类型。

```
//日期到字符串的转换
String today = df.format(new Date());
//字符串到日期的转换
Date date = df.parse("2009-06-12 02:06:37");
```

8.6.3 SimpleDateFormat类

3．应用举例

(1)获得距离指定日期n天的那一天的日期。

(2) 获得给定两个日期相差度天数。

参见DateClass2.java

本章小结

本章对Java中的常用类：String、StringBuffer、StringTokenizer、Math、基本数据类型的包装类、日期日历类进行了介绍。

字符串在程序中经常使用，要区分字符串常量与调用构造器生成的字符串对象的不同之处。字符串的一个重要特性是其内容的不可改变性，若要修改其内容，就会产生新对象，当这样的操作很频繁时就会影响系统运行效率，这一问题可由StringBuffer类来解决。String类的构造器有多个，它们适用于不同的场合；提供的方法较多，使用灵活，要逐一掌握不是一件容易的事情，可按功能进行分类、结合单词的拼写、一些简单例子，掌握典型用法即可。

StringBuffer类可以弥补String类的不足，其内容可以任意修改，进行增加、删除、修改、查询是它的主要操作，当存放的内容超过StringBuffer的容量时，它能自动扩展。从JDK 5.0起还新增了一个与StringBuffer功能相同的类——StringBuilder，它执行效率更高，但不具备多线程下的安全性。

本章小结

字符串中可能包含一些分隔符，当要进一步获取各分隔部分内容时，可使用StringTokenizer类。该类在构造时可使用默认的分隔符，也可自己指定分隔符，数量可以为一个或多个，利用循环可逐一得到字符串的各组成元素。所提供的方法有两种不同的组合：hasMoreTokens()和nextToken()一组、hasMoreElements()和nextElement()一组，调用时选择其中一组即可。

Math类的主要功能是计算，包括绝对值、平方根、指数、对数、随机数、三角函数与反三角函数值等运算，也可获得e和 的近似值。不过，这些属性与方法都是静态(static)的，调用格式为：Math.属性 或 Math.方法([参数表]) 。

本章小结

引入包装类的目的是为基本类型数据提供对象类型，8个包装类的名称很容易记忆。包装类提供了一些“共性”的常量、构造方法和常用方法，请予以理解，重点掌握三类方法：包装类对象与基本类型数据的相互转换、包装类对象与字符串的相互转换、基本类型数据与字符串的相互转换。

日期日历类提供了时间操作功能，通常用距离GMT 1970年1月1日00时00分00秒的毫秒数来表示时间。Date类具有操作时间的基本功能，但由于在设计上存在严重缺陷，它的多个方法已过时、废弃，相关功能已转移到日历类中实现；GregorianCalendar类是日历类的一个具体子类，它定义了多个字段常量，通过get (int field)、set(int field, int val)方法可得到、设置时间分量的值。SimpleDateFormat类为用户提供了通过定义“输出模式”来控制日期输出格式的方法。