Java中的Date类和Calendar类

1. **Date类 直接继承于java.lang.Object类，存在于java.util 工具包中。Date类没有字段（Fields），Calendar类提供了特别多的字段。**Date类中的很多构造方法和成员方法都已经过时了，Deprecated 表示弃用的，过时的。
2. **Date类实现了Comparab<Date>接口，则Date类肯定会有compareTo（Date anotherDate）方法，返回int值，只有0，+1，-1。如果返回0，则两个Date对象表示同一个时刻；如果返回1，则调拥此方法的对象对应的时间比参数对象晚，如果返回-1，则是调用此方法的Date对象比参数Date对象早。**



如：

Date date = new Date();

Thread.sleep(2000);

Date date2 = new Date();

System.out.println(date.compareTo(date2));// 返回-1

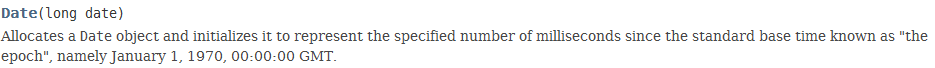
System.out.println(date2.compareTo(date));// 返回 1

System.out.println(date2.compareTo(date2));// 返回 0，自己和自己比总是返回0

1. **Java中年月日时分秒的存在形式：**
2. **年份year由整数year-1900表示；**
3. **月份从0-11；（注意 0代表1月，11代表12月，12代表下一年1月）**
4. **日期（一月中某一天）用1-31表示；例子：1月32日表示2月1号**
5. **小时用0-23表示；**
6. **分钟用0—59表示；**
7. **秒用0-61表示，其中60,61用于表示闰秒。**
8. **Date类的构造方法：**
9. **Date（）：把当前的时间封装成Date对象；**



1. **Date（long millisecond）：把指定的毫秒值封装成Date对象；**



**注 : 毫秒值（millisecond）**是从1970年1月1号00:00:00 GMT（时区）开始计算的。

1. **Date 类的成员方法：**
2. **getXxx（）方法：只可以获取毫秒值，其他的年月日时分秒等值需要通过Calendar中的属性和方法获取。**
   1. **getTime（）方法：获取此对象的毫秒值，毫秒值都是利用long类型的。**



* 1. **对于getDate、getTimezoneOffset、getYear、getDay、getMonth、getHours、getMinutes、getSeconds等方法都已经被Calendar中的方法Calendar.get（Calendar.XXXX）取代。**

1. **setXxx（）方法：修改Date对象的时间，只可以通过毫秒值修改，其他方法已经过时。**
   1. **setTime（long time）方法：重新设置时间，返回为void。**

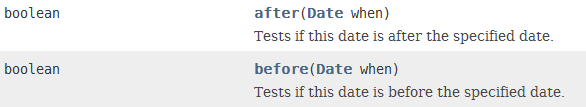


* 1. **其他如同gerXxx一样，已经被Calendar.set（Calendar.XXX , xxx）取代了。**

1. **比较方法：**
2. **compareTo（Date anotherDate）：返回int值 ： 0，-1，+1。（见上面介绍）**
3. **equals（Object obj）：返回boolean值，比较的是时间，已经被覆写了。**



1. **after（Date anotherDate）方法：返回的是boolean值。**
2. **before（Date anotherDate）方法：返回的是boolean值。**



**注意：**对于after和before方法：如果自己和自己比较返回的是false。

1. **toString（）方法：**



**如：Date date = new Date();**

**System.out.println(date.toString());//结果如下：**



1. **from（）方法：静态方法，从一个Instant对象中获取一个Date对象。**



1. **毫秒值和Date对象之间的转换：**
2. **毫秒值-🡪Date对象：**
   1. **利用构造方法：Date（long millisecond），**即可以将此毫秒值转成日期对象；
   2. **利用setTime（long time）方法：将一个已经存在的Date对象的值修改为指定的毫秒值时间，返回值为void。**
3. **Date对象---🡪毫秒值：**

**通过Date类的getTime（）方法，无参数，返回一个long类型的millisecond值。**

1. **毫秒值和Date日期对象相互转换的意义：**
2. **毫秒值-🡪Date对象**

**意义很明显，仅仅知道毫秒值不能很明显知道年月日时分秒，通过封装成Date对象后（加上Calendar功能），就可以很方便可以利用Date 对象的各种方法对日期时间的各个字段（年月日时分秒）进行操作。**

1. **Date对象---🡪毫秒值：**

日期对象转成毫秒值的意义：**可以通过毫米值进行加减计算**，如获取两个日期键的天数、小时数等。后面有个小实例。

1. **日期对象和字符串之间的转换：**
2. **日期对象🡺字符串（即日期对象的格式化）：**
   1. **Date类的toString（）方法给了一种默认的日期输出格式 （转换成字符串输出）。**
   2. **但是由于不同国家存在差异，所以有必要提供更多种格式及让程序员指定特定的日期格式。基于此需求，出现了DateFormat类。**

**利用DateFormat类中的format（Date date）方法，就可以把日期对象转换成相应的字符串。调用format方法需要先有getXXXInstance方法获得具有int style : 有DEFAULT、SHORT、LONG、FULL、MEDIUM样式的DateFormat对象。**

**如:** **String str = DateFormat.getDateInstance(DateFormat.LONG).format(date);**

1. **字符串🡺日期对象：**
   * 1. **利用DateFormat类中的parse（String source），返回的是一个Date对象。**
2. **静态工厂方法：**

**Java中用于产生对象的方法称为工厂方法。**

1. **单例设计模式：生成一个对象；**
2. **这里通过调用format方法：对一个Date对象进行设定特定格式，也属于工厂方法。如：myString = DateFormat.getDateInstance().format(myDate);**
3. **DateFormat类：存在于java.text包（文本包）中，直接继承于java.text.Format类，是一个抽象类（abstract类），不可直接实例化，但是可以通过静态工厂方法即通过静态方法创建DateFormat对象。DateFormat有一个直接实现子类SimpleDateFormat,这个子类SimpleDateFormat主要用于自定义日期格式。**
4. **DateFormat提供了很多的字段值，都是静态的：**
5. **格式样式字段值：都是静态的，int类型的。**

**DEFAULT、SHORT、LONG、FULL、MEDIUM等样式，都是int类型的。**

**在DateFormat的静态工厂方法中用到。这是java提供的，如果想要自定义可以利用子类SimpleDateFormat定义。**

1. **年月日时分秒字段值：DATA\_FIELD等**
2. **DateFormat的主要方法介绍：**
3. **静态工厂方法：返回的都是一个DateFormat对象，都是静态方法。**

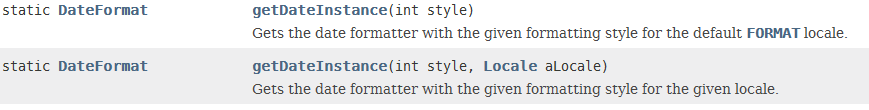
**int style : 有DEFAULT、SHORT、LONG、FULL、MEDIUM。**

* + 1. **getInstance（）方法：**



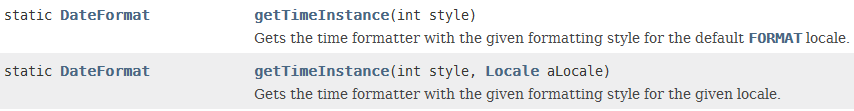
* + 1. **getDateInstance（）方法：只有年月日。**





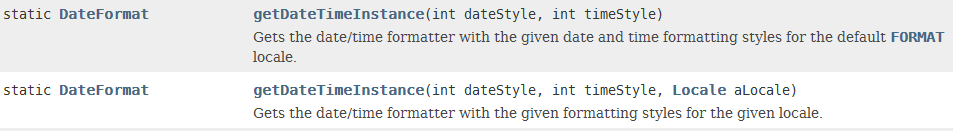
* + 1. **getTimeInstance()方法:只有时分秒。**





* + 1. **getDateTimeInstance()方法：年月日时分秒都有。**





**示例1：**

Date date = new Date();

DateFormat df0 = DateFormat.getInstance();

DateFormat df1 = DateFormat.getDateInstance();

DateFormat df2 = DateFormat.getTimeInstance();

DateFormat df3 = DateFormat.getDateTimeInstance();

String str0 = df0.format(date);//16-1-2 下午2:36

String str1 = df1.format(date);//2016-1-2

String str2 = df2.format(date);//14:36:11

String str3 = df3.format(date);//2016-1-2 14:36:11

示例2：以**getDateTimeInstance（）方法**为例：

Date date = new Date();

DateFormat df1 = DateFormat.getDateTimeInstance(DateFormat.DEFAULT,DateFormat.DEFAULT);

DateFormat df2 = DateFormat.getDateTimeInstance(DateFormat.LONG,DateFormat.LONG);

DateFormat df3 = DateFormat.getDateTimeInstance(DateFormat.SHORT,DateFormat.SHORT);

DateFormat df4 = DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL);

DateFormat df5 = DateFormat.getDateTimeInstance(DateFormat.MEDIUM,DateFormat.MEDIUM);

String str1 = df1.format(date);**//2016-1-2 14:48:22 DEFAULT**

String str2 = df2.format(date);**//2016年1月2日 下午02时48分22秒** LONG

String str3 = df3.format(date);**//16-1-2 下午2:48**  SHORT

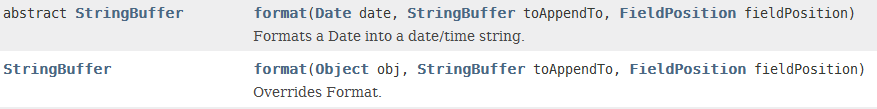
String str4 = df4.format(date);**//2016年1月2日 星期六 下午02时48分22秒 CST** FULL

String str5 = df5.format(date);**//2016-1-2 14:48:22**  MEDIUM

1. **format（）方法：**
   1. **format（Date date）：返回一个字符串。**



* 1. **另外两个：**



1. **非静态工厂方法：**
   1. **parse(String source)方法：将字符串转换成Date对象。**

**注意：parse有可能会抛出ParseException异常。**



* 1. **getCalendar（），获取一个Calendar对象。**



* 1. **getNumberFormat（）：**



* 1. **getTimeZone():**



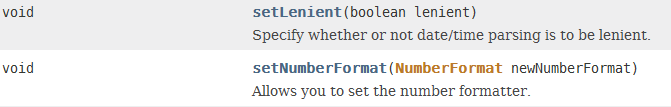
1. **setXxx方法：**
   1. **setCalendar（Calendar newCalendar）方法：**



* 1. **setTimeZone（TimeZone zone）方法：**



* 1. **setXXXX方法：**



1. **注意DateFormat的toString（）是Object的，没有对其覆写。**
2. **SimpleDateFormat的构造方法：**
3. **空参数的构造方法：利用的是默认的格式并使用默认的日期格式符号集。**



1. **指定日期样式：采用默认的日期格式符号集。**



1. **指定日期样式和日期格式符号集：**

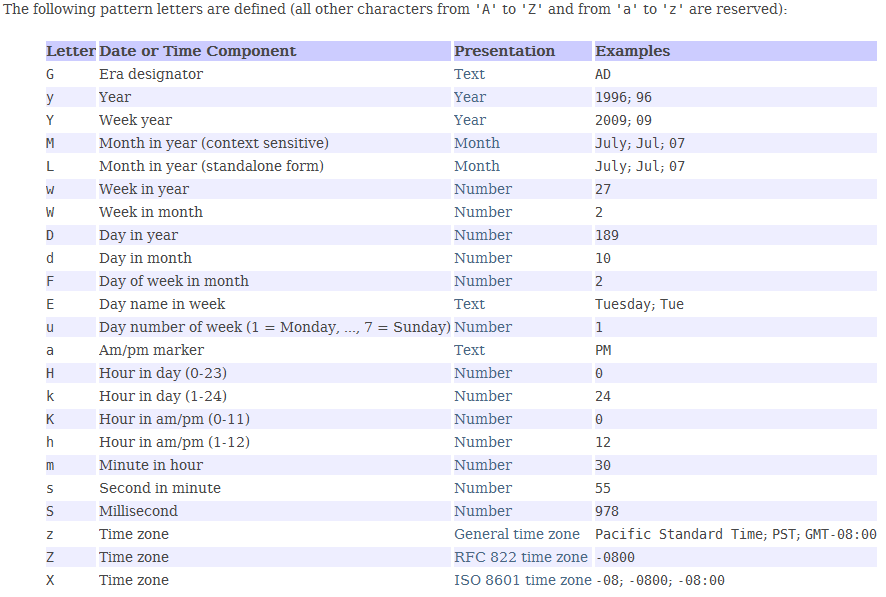


1. **指定日期样式和时区：**

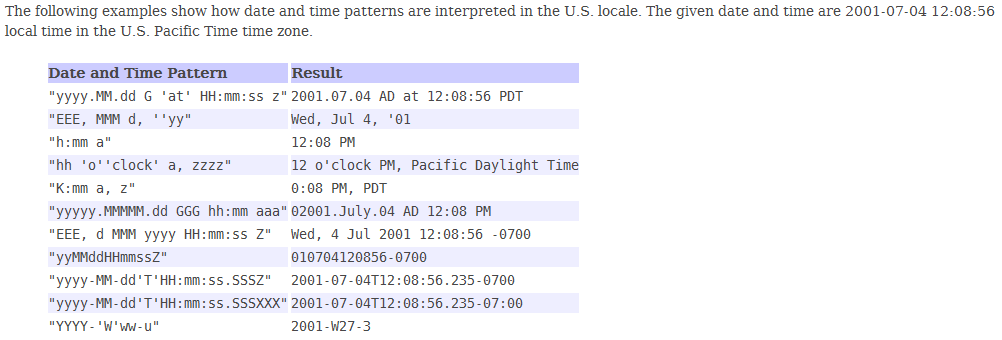


1. **SimpleDateFormat类用于自定义格式：**

**用特定的大写和小写字母表示年月日时分秒等字段，利用双引号括起来构成一个整体，如果需要添加固定的文本，可以利用单括号在内部括起来。见示例。**



**示例：**



**示例：自定义格式:**

**DateFormat df = new SimpleDateFormat("yyyy年.MM月.dd日 .hh时.mm分");**

**String str = df.format(new Date());//2016年.01月.02日 .03时.25分**

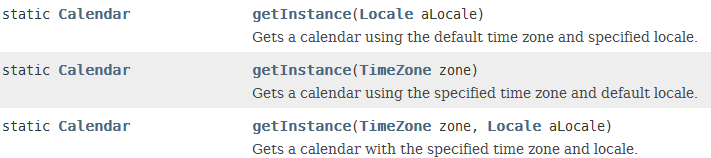
1. **Date类由于不易实现国际化，部分功能被Calendar抽象类取代。**
2. **Calendar类是抽象类，存在于java.util工具包中，直接继承于java.lang.Object类。采用对象单例化，静态工厂方法是getInstance（）方法。**

**Calendar提供了特别多的字段，很多能想到的都有了。**

1. **Calendar虽然是抽象类，但是拥有构造方法供子类使用。**
2. **Calendar类的成员方法：**
3. **静态工厂方法：**
   1. **getInstance（）方法：返回一个Calendar对象；**



* 1. **getInstance（XXXXX）方法：**



1. **非静态工厂方法：**
   * 1. **getTime（）方法： 返回一个Date对象；**



* + 1. **getTimeZone（）方法：返回一个TimeZone对象；**



* + 1. **getInstant（）方法： 返回一个Instant对象；**



1. **toString（）方法：把所有的属性都以键值对的形式打印出来了。**



**打印结果如下：**

java.util.GregorianCalendar[time=1451722167076,areFieldsSet=true,areAllFieldsSet=true,lenient=true,zone=sun.util.calendar.ZoneInfo[id="Asia/Shanghai",offset=28800000,dstSavings=0,useDaylight=false,transitions=19,lastRule=null],firstDayOfWeek=1,minimalDaysInFirstWeek=1,ERA=1,YEAR=2016,MONTH=0,WEEK\_OF\_YEAR=1,WEEK\_OF\_MONTH=1,DAY\_OF\_MONTH=2,DAY\_OF\_YEAR=2,DAY\_OF\_WEEK=7,DAY\_OF\_WEEK\_IN\_MONTH=1,AM\_PM=1,HOUR=4,HOUR\_OF\_DAY=16,MINUTE=9,SECOND=27,MILLISECOND=76,ZONE\_OFFSET=28800000,DST\_OFFSET=0]

1. **get方法：**

**a） get（int field）方法：返回int值，可以把所有的字段打印出来了。**



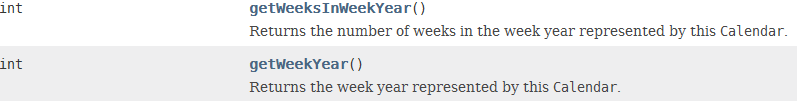
**注意：打印出类的月份需要加1才可以。**

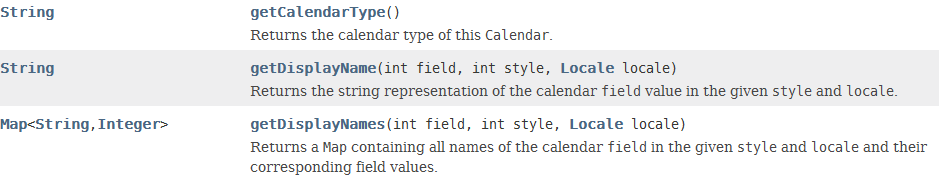
**如：**Calendar c = Calendar.getInstance();

System.out.println(c.get(Calendar.YEAR));//2016

System.out.println(c.get(Calendar.MONTH)+1);//1

b) getXXX()方法：



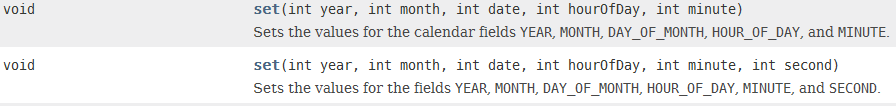


1. **set()方法：利用方法重载来实现不同的功能。**
   * 1. **set（int field，int value）方法：**



* + 1. **set（int year，int month，int day）方法：**

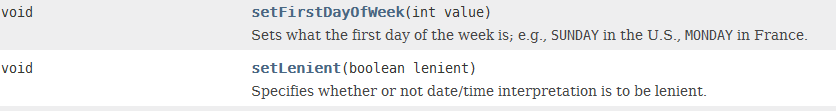


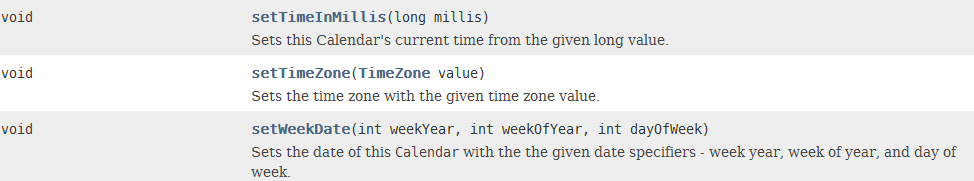


* + 1. **setTime（Date date）方法：**



* + 1. **setXxx方法：**

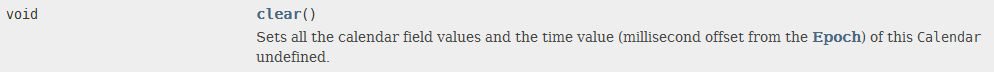


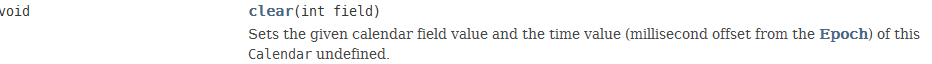


1. **add（int field，int amount）方法：**



1. **clear（）方法 和 clear（int field）方法：**

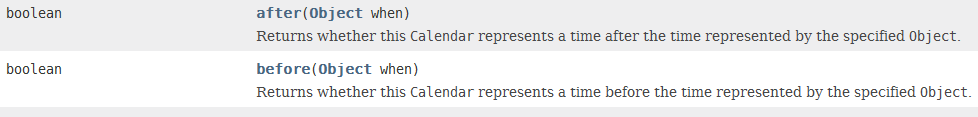


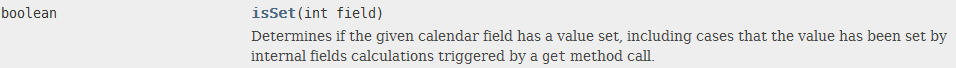


1. **判断方法：同Date类一样。**









1. **还有一些不常用到，不再赘述。**
2. **Date类和Calendar类：**
3. **这两个类仍然都在使用；Date类中的很多方法淘汰了，但是Date类仍然广泛使用。**
4. **Date类中没有提供字段值，不可以获取某个字段值；而Calendar类提供了很多字段值，并且有很多get和set方法，进行获取和设置这些字段，同时有删除clear方法等；**
5. **Date类可以直接实例化，而Calendar是个抽象类，不可以直接实例化，通过静态工厂方法getInstance获取对象。**
6. **DateFormat是对Date对象进行加工的，目前不能对Calendar进行加工。**
7. **练习题：**
   1. 将**日期格式的字符串**转换成**日期对象:**

**思路：**

* + 1. 首先根据日期字符串的格式，生成一个符合字符串格式的DateFormat对象或者SimpleDateFormat对象；
    2. 利用DateFormat对象或者SimpleDateFormat对象的parse方法，将字符串转换成**Date对象**。

**代码如下：**

String str\_date1 = new String("2011-9-11");

String str\_date2 = new String("2015年3月5日");

String str\_date3 = new String("2015----12----24");

//str\_data1为DEFAULT格式，构造日期格式器

DateFormat df1 = DateFormat.*getDateInstance*(DateFormat.*DEFAULT*);

//str\_data2为LONG格式，构造日期格式器

DateFormat df2 = DateFormat.*getDateInstance*(DateFormat.*LONG*);

//str\_data3为自定义格式，构造日期格式器

DateFormat df3 = new SimpleDateFormat("yyyy----MM----dd");

Date date1 = df1.parse(str\_date1);**//Sun Sep 11 00:00:00 CST 2011**

Date date2 = df2.parse(str\_date2);**//Thu Mar 05 00:00:00 CST 2015**

Date date3 = df3.parse(str\_date3);**//Thu Dec 24 00:00:00 CST 2015**

* 1. **计算两个日期之间还有多少天？**

**日期是给出字符串形式 2016年1月2日 和**  **2015年10月1日**

**思路：**

1. 把**日期值**转换成**毫秒值**，从而可以进行**加减计算**。
2. 首先将**日期字符串**转换成**日期对象**，利用相应的日期格式器的parse方法；
3. 再把**日期对象**转换成**毫秒值**，利用Date的getTime（）方法；
4. 通过加减计算**即可计算**。

代码：

String str1 = new String("2016年5月2日");

String str2 = new String("2015年5月1日");

DateFormat df = new SimpleDateFormat("yyyy年MM月dd日");

Date date1 = df.parse(str1);

Date date2 = df.parse(str2);

long time1 = date1.getTime();

long time2 = date2.getTime();

int day =(int)((time1-time2)/1000/60/60/24);

System.out.println("两个日期间隔 ："+day);**// 两个日期间隔 ：367**

* 1. 使用Calendar类获取、设置、删除一些字段值等操作。

示例： Calendar c = Calendar.getInstance();

System.out.println(c.get(Calendar.YEAR));//2016

System.out.println(c.get(Calendar.MONTH)+1);//1

c.set(2017,13,1);

System.out.println(c.get(Calendar.YEAR));//2018

System.out.println(c.get(Calendar.MONTH)+1);//2

System.out.println(c.isSet(Calendar.YEAR));//true

c.clear(Calendar.YEAR);

System.out.println(c.isSet(Calendar.YEAR));//false