

## 以下内容节选自 Java 私塾自编经典教材

Java 中经常会使用到日期操作的类, 下面详细的讲述一下。

## 六: Java 日期操作的类

---

### 1: Date 类

---

java.util 包里面的 Date 类, 是 Java 里面进行日期操作常用类。Date 类用来表示特定的瞬间, 精确到毫秒。

#### 1.1: 如何初始化 Date

##### 构造方法: Date()

分配 Date 对象并初始化此对象, 以表示分配它的时候的当前时间 (精确到毫秒)。使用 Date 类得到当前的时间。

##### 构造方法: Date(long date)

分配 Date 对象并初始化此对象, 以表示自从标准基准时间 (称为“历元 (epoch)”, 即 1970 年 1 月 1 日 00:00:00 格林威治时间) 以来的指定毫秒数。

#### 2.2: 常用方法

##### 方法: after(Date when)

测试此日期是否在指定日期之后

##### 方法: before(Date when)

测试此日期是否在指定日期之前

##### 方法: getTime()

返回自 1970 年 1 月 1 日 00:00:00 GMT 以来此 Date 对象表示的毫秒数。

#### 2.3: 示例: 简单的性能测试——监控一段代码运行所需要的时间

```
public class Test {  
    public static void main(String args[]) {  
        long d1 = new Date().getTime();//得到此时的时间  
        int sum = 0;  
        for(int i=1;i<=1000000;i++){//看清楚了, 可是一百万次  
            sum +=i;  
        }  
        System.out.println("从1加到1000000的和="+sum);  
        long d2 = new Date().getTime();//得到此时的时间  
        System.out.println("从1加到1000000所耗费的时间是="+ (d2-d1) + "毫秒");  
    }  
}
```

运行结果:

从1加到1000000的和=1784293664

从 1 加到 1000000 所耗费的时间是=20 毫秒

## 2: DateFormat 类和 SimpleDateFormat 类

在 java.text 包中的 DateFormat 类, 是日期/时间格式化子类的抽象类, 它以与语言无关的方式格式化并解析日期或时间。日期/时间格式化子类 (如 SimpleDateFormat) 允许进行格式化 (也就是日期——>文本)、解析 (文本——> 日期) 和标准化。

由于 DateFormat 是个抽象类, SimpleDateFormat 类是它的子类, 所以下面就主要按照 SimpleDateFormat 类来讲解。

### 2.1: 如何初始化

这里只讲述最常用到的构造方法, 更多的请参看 JDK 文档。

**构造方法: SimpleDateFormat(String pattern)**

用给定的模式和默认语言环境的日期格式符号构造 SimpleDateFormat

### 2.2 日期和时间模式

日期和时间格式由 *日期和时间模式* 字符串指定。在日期和时间模式字符串中, 未加引号的字母 'A' 到 'Z' 和 'a' 到 'z' 被解释为模式字母, 用来表示日期或时间字符串元素。文本可以使用单引号 (') 引起来, 以免进行解释。''' 表示单引号。所有其他字符均不解释; 只是在格式化时将它们简单复制到输出字符串, 或者在解析时与输入字符串进行匹配。

定义了以下模式字母 (所有其他字符 'A' 到 'Z' 和 'a' 到 'z' 都被保留):

字母	日期或时间元素	表示	示例
G	Era	标志符	Text
y	年	Year	1996;
M	年中的月份	Month	July;
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;
a	Am/pm	标记	Text
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
h	am/pm	中的小时数 (1-12)	Number
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
Z	时区	RFC	822

### 2.3: 常用方法

方法: `parse(String source)`

从给定字符串的开始解析文本, 以生成一个日期

方法: `format(Date date)`

将一个 `Date` 格式化为日期/时间字符串

这里只讲述最常用的方法, 更多的方法请参看 JDK 文档。

### 2.4: 示例

```
import java.util.*;
import java.text.*;

public class Test {
    public static void main(String args[]) {
        DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss SSS");

        Date d = new Date();
        //把当前时间转换成为我们熟悉的时间表达格式
        String str = df.format(d);

        System.out.println("当前时间是: "+str);

        //然后再把字符串格式的日期转换成为一个Date类
        try {
            Date d2 = df.parse("2008-08-08 08:08:08 888");
            System.out.println("北京奥运会开幕时间是: "+d2.getTime());
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
}
```

运行结果:

当前时间是: 2008-07-22 00:57:45 612

北京奥运会开幕时间是: 1218154088888

### 2.5: 说明

虽然 JDK 文档上说 `Date` 的毫秒值, 是相对于格林威治时间 1970 年 1 月 1 号的 0 点, 但实际测试, 这个 `Date` 是跟时区相关的, 也就是说在中国测试这个基准值应该是 1970 年 1 月 1 日的 8 点, 不过这个不影响我们的处理, 因为只要是同一个基准时间就可以了, 而不用担心具体是多少, 见下面的示例:

```
public class Test {
    public static void main(String args[]) {
        DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss SSS");

        Date d = new Date(0L); //把时间设为0, 表示到基准时间
        //然后转换成为字符串看看是什么时候
        String str = df.format(d);
    }
}
```

```
        System.out.println("基准时间是: "+str);
    }
}
```

运行结果:

基准时间是: 1970-01-01 08:00:00 000

### 3: Calendar 类

---

java.util 包中的 Calendar 类是 Java 里面另外一个常用的日期处理的类。Calendar 类是一个抽象类, 它为特定瞬间与一组诸如 YEAR、MONTH、DAY\_OF\_MONTH、HOUR 等 日历字段之间的转换提供了一些方法, 并为操作日历字段(例如获得下星期的日期)提供了一些方法。

#### 3.1: 如何初始化

Calendar 类是通过一个静态方法 getInstance() 来获取 Calendar 实例。返回的 Calendar 基于当前时间, 使用了默认时区和默认语言环境

如下: Calendar c = Calendar.getInstance();

#### 3.2: 使用 Calendar 对日期进行部分析取

Calendar 类一个重要的功能就是能够从日期里面按照要求析取出数据, 如: 年、月、日、星期等等。

方法: get(int field)

返回给定日历字段的值

示例如下:

```
public class Test {
    public static void main(String args[]) {
        Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH); //注意: month特殊, 是从0开始的,
        也就是0表示1月
        int day = c.get(Calendar.DAY_OF_MONTH);

        System.out.println("现在是"+year+"年"+(month+1)+"月"+day+"日");
    }
}
```

运行结果:

现在是 2008 年 7 月 22 日

#### 3.3: 使用 Calendar 进行日期运算

这是 Calendar 另外一个常用的功能, 也就是对日期进行加加减减的运算。

方法: add(int field, int amount)

根据日历的规则, 为给定的日历字段添加或减去指定的时间量

示例如下:

```
public class Test {
    public static void main(String args[]) {
        Calendar c = Calendar.getInstance();
```

```
c.add(Calendar.DATE, 12); //当前日期加12天, 如果是-12就表示当前日期减  
去12天
```

```
int year = c.get(Calendar.YEAR);  
int month = c.get(Calendar.MONTH); //注意: month特殊, 是从0开始的,  
也就是0表示1月
```

```
int day = c.get(Calendar.DAY_OF_MONTH);
```

```
System.out.println("在当前日期加12天是"+year+"年"+(month+1)+"月  
"+day+"日");  
}
```

```
}
```

运行结果: 在当前日期加 12 天是 2008 年 8 月 3 日

### 3.4: 为 Calendar 设置初始值

方法 setTime(Date date)

使用给定的 Date 设置此 Calendar 的当前时间

方法 setTimeInMillis(long millis)

用给定的 long 值设置此 Calendar 的当前时间值

```
public class Test {
```

```
    public static void main(String args[]) {
```

```
        Calendar c = Calendar.getInstance();
```

```
        c.setTimeInMillis(1234567890123L);
```

```
        int year = c.get(Calendar.YEAR);
```

```
        int month = c.get(Calendar.MONTH); //注意: month特殊, 是从0开始的,  
也就是0表示1月
```

```
        int day = c.get(Calendar.DAY_OF_MONTH);
```

```
        System.out.println("设置的时间是"+year+"年"+(month+1)+"月"+day+"  
日");  
    }
```

```
}
```

运行结果: 设置的时间是 2009 年 2 月 14 日