[java(Date,Calendar类)学习笔记](http://blog.csdn.net/shanliangliuxing/article/details/7327445)

**9.6 java.util包**

         java.util包是Java语言提供的工具类包，该包中包含了如日期、随机数和数据结构实现类等一系列的工具类实现，是学习Java语言的基础包之一。

         本部分就以Java语言中常用的时间和日期处理、随机数处理以及集合框架为基础进行介绍。

**9.6.1 时间和日期处理**

         使用程序进行时间和日期处理，是程序员必须的一种常用技能，在不同的程序设计语言中提供了不同的格式进行实现，现在就介绍一下Java语言中的实现方式，以及基本的应用，使得对于Java语言的时间和日期处理技术有比较全面的认识。

         在程序中，某个固定的时间代表的都是一个时间点，也就是一个时间的瞬间，例如2009年3月8日15点50分0秒，在实际的应用中，经常需要对于两个时间进行比较或计算时间之间的差值，这些功能在Java语言中都可以很方便的实现。

         在Java语言中，时间的表达单位是毫秒。也就是说，Java语言中的时间处理可以精确到毫秒。

         在Java语言中，表达时间的方式有两种：

a、绝对时间。以直观的形式来表达某个时间点，例如2009年10月10号0点0分0秒。使用这种形式表达时间，使用起来比较直观，但是不方便进行时间之间的计算。例如无法很直观的计算2020年1月1号0点0分0秒和上面这个时间之间相差多少天。绝对时间以对象的形式进行表达，Java API中提供了java.util包中的Date类和Calendar类的对象进行表达。

b、相对时间。以一个long型的数字表达某个时间点。例如102847423468。使用这种方式的优缺点和绝对时间刚好相反。这种方式很方便时间之间的计算，但是阅读起来很不直观。在Java API中以需要表达的时间点，例如2009年10月10号0点0分0秒，和GMT(格林威治时间，也就是伦敦时间)1970年1月1号0点0分0秒之间相差的毫秒数作为相对时间的数值，如果该时间在这个时间只好，则相对时间为正数，否则相对时间为负数。Java API中提供了java.lang包中的System类的currentTimeMillis方法，获得以相对时间形式描述的当前系统时间。

在实际使用时，绝对时间和相对时间之间可以很方便的进行转换。

**9.6.1.1 Date类**

在JDK1.0中，Date类是唯一的一个代表时间的类，但是由于Date类不便于实现国际化，所以从JDK1.1版本开始，推荐使用Calendar类进行时间和日期处理。这里简单介绍一下Date类的使用。

1、使用Date类代表当前系统时间

                   Date d = new Date();

                   System.out.println(d);

使用Date类的默认构造方法创建出的对象就代表当前时间，由于Date类覆盖了toString方法，所以可以直接输出Date类型的对象，显示的结果如下：

                            Sun Mar 08 16:35:58 CST 2009

在该格式中，Sun代表Sunday(周日)，Mar代表March(三月)，08代表8号，CST代表China Standard Time(中国标准时间，也就是北京时间(东八区))。

         2、使用Date类代表指定的时间

                            Date d1 = new Date(2009-1900,3-1,9);

                            System.out.println(d1);

使用带参数的构造方法，可以构造指定日期的Date类对象，Date类中年份的参数应该是实际需要代表的年份减去1900，实际需要代表的月份减去1以后的值。例如上面的示例代码代表就是2009年3月9号。

实际代表具体的年月日时分秒的日期对象，和这个类似。

         3、获得Date对象中的信息

                            Date d2 = new Date();

                   //年份

                   int year = d2.getYear() + 1900;

                   //月份

                   int month = d2.getMonth() + 1;

                   //日期

                   int date = d2.getDate();

                   //小时

                   int hour = d2.getHours();

                   //分钟

                   int minute = d2.getMinutes();

                   //秒

                   int second = d2.getSeconds();

                   //星期几

                   int day = d2.getDay();

                   System.out.println("年份：" + year);

                   System.out.println("月份：" + month);

                   System.out.println("日期：" + date);

                   System.out.println("小时：" + hour);

                   System.out.println("分钟：" + minute);

                   System.out.println("秒：" + second);

                   System.out.println("星期：" + day);

         使用Date类中对应的get方法，可以获得Date类对象中相关的信息，需要注意的是使用getYear获得是Date对象中年份减去1900以后的值，所以需要显示对应的年份则需要在返回值的基础上加上1900，月份类似。在Date类中还提供了getDay方法，用于获得Date对象代表的时间是星期几，Date类规定周日是0，周一是1，周二是2，后续的依次类推。

4、Date对象和相对时间之间的互转

                  Date d3 = new Date(2009-1900,3-1,10);

                   long time = 1290876532190L;

                   //将Date类的对象转换为相对时间

                   long t = d3.getTime();

                   System.out.println(t);

                   //将相对时间转换为Date类的对象

                   Date d4 = new Date(time);

                   System.out.println(d4);

         使用Date对象中的getTime方法，可以将Date类的对象转换为相对时间，使用Date类的构造方法，可以将相对时间转换为Date类的对象。经过转换以后，既方便了时间的计算，也使时间显示比较直观了。

**9.6.1.2 Calendar类**

从JDK1.1版本开始，在处理日期和时间时，系统推荐使用Calendar类进行实现。在设计上，Calendar类的功能要比Date类强大很多，而且在实现方式上也比Date类要复杂一些，下面就介绍一下Calendar类的使用。

Calendar类是一个抽象类，在实际使用时实现特定的子类的对象，创建对象的过程对程序员来说是透明的，只需要使用getInstance方法创建即可。

1、使用Calendar类代表当前时间

                   Calendar c = Calendar.getInstance();

由于Calendar类是抽象类，且Calendar类的构造方法是protected的，所以无法使用Calendar类的构造方法来创建对象，API中提供了getInstance方法用来创建对象。

使用该方法获得的Calendar对象就代表当前的系统时间，由于Calendar类toString实现的没有Date类那么直观，所以直接输出Calendar类的对象意义不大。

2、使用Calendar类代表指定的时间

                   Calendar c1 = Calendar.getInstance();

                   c1.set(2009, 3 - 1, 9);

使用Calendar类代表特定的时间，需要首先创建一个Calendar的对象，然后再设定该对象中的年月日参数来完成。

set方法的声明为：

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

以上示例代码设置的时间为2009年3月9日，其参数的结构和Date类不一样。Calendar类中年份的数值直接书写，月份的值为实际的月份值减1，日期的值就是实际的日期值。

如果只设定某个字段，例如日期的值，则可以使用如下set方法：

         public void set(int field,int value)

在该方法中，参数field代表要设置的字段的类型，常见类型如下：

         Calendar.YEAR——年份

         Calendar.MONTH——月份

         Calendar.DATE——日期

         Calendar.DAY\_OF\_MONTH——日期，和上面的字段完全相同

         Calendar.HOUR——12小时制的小时数

         Calendar.HOUR\_OF\_DAY——24小时制的小时数

         Calendar.MINUTE——分钟

         Calendar.SECOND——秒

         Calendar.DAY\_OF\_WEEK——星期几

后续的参数value代表，设置成的值。例如：

         c1.set(Calendar.DATE,10);

该代码的作用是将c1对象代表的时间中日期设置为10号，其它所有的数值会被重新计算，例如星期几以及对应的相对时间数值等。

3、获得Calendar类中的信息

                   Calendar c2 = Calendar.getInstance();

                   //年份

                   int year = c2.get(Calendar.YEAR);

                   //月份

                   int month = c2.get(Calendar.MONTH) + 1;

                   //日期

                   int date = c2.get(Calendar.DATE);

                   //小时

                   int hour = c2.get(Calendar.HOUR\_OF\_DAY);

                   //分钟

                   int minute = c2.get(Calendar.MINUTE);

                   //秒

                   int second = c2.get(Calendar.SECOND);

                   //星期几

                   int day = c2.get(Calendar.DAY\_OF\_WEEK);

                   System.out.println("年份：" + year);

                   System.out.println("月份：" + month);

                   System.out.println("日期：" + date);

                   System.out.println("小时：" + hour);

                   System.out.println("分钟：" + minute);

                   System.out.println("秒：" + second);

                   System.out.println("星期：" + day);

使用Calendar类中的get方法可以获得Calendar对象中对应的信息，get方法的声明如下：

         public int get(int field)

其中参数field代表需要获得的字段的值，字段说明和上面的set方法保持一致。需要说明的是，获得的月份为实际的月份值减1，获得的星期的值和Date类不一样。在Calendar类中，周日是1，周一是2，周二是3，依次类推。

4、其它方法说明

其实Calendar类中还提供了很多其它有用的方法，下面简单的介绍几个常见方法的使用。

a、add方法

                   public abstract void add(int field,int amount)

该方法的作用是在Calendar对象中的某个字段上增加或减少一定的数值，增加是amount的值为正，减少时amount的值为负。

         例如在计算一下当前时间100天以后的日期，代码如下：

                   Calendar c3 = Calendar.getInstance();

                   c3.add(Calendar.DATE, 100);

                   int year1 = c3.get(Calendar.YEAR);

                   //月份

                   int month1 = c3.get(Calendar.MONTH) + 1;

                   //日期

                   int date1 = c3.get(Calendar.DATE);

                   System.out.println(year1 + "年" + month1 + "月" + date1 + "日");

这里add方法是指在c3对象的Calendar.DATE，也就是日期字段上增加100，类内部会重新计算该日期对象中其它各字段的值，从而获得100天以后的日期，例如程序的输出结果可能为：

                   2009年6月17日

b、after方法

                   public boolean after(Object when)

该方法的作用是判断当前日期对象是否在when对象的后面，如果在when对象的后面则返回true，否则返回false。例如：

                   Calendar c4 = Calendar.getInstance();

                   c4.set(2009, 10 - 1, 10);

                   Calendar c5 = Calendar.getInstance();

                   c5.set(2010, 10 - 1, 10);

                   boolean b = c5.after(c4);

                   System.out.println(b);

在该示例代码中对象c4代表的时间是2009年10月10号，对象c5代表的时间是2010年10月10号，则对象c5代表的日期在c4代表的日期之后，所以after方法的返回值是true。

另外一个类似的方法是before，该方法是判断当前日期对象是否位于另外一个日期对象之前。

                   c、getTime方法

                                     public final Date getTime()

该方法的作用是将Calendar类型的对象转换为对应的Date类对象，两者代表相同的时间点。

类似的方法是setTime，该方法的作用是将Date对象转换为对应的Calendar对象，该方法的声明如下：

         public final void setTime(Date date)

转换的示例代码如下：

                  Date d = new Date();

                   Calendar c6 = Calendar.getInstance();

                   //Calendar类型的对象转换为Date对象

                   Date d1 = c6.getTime();

                   //Date类型的对象转换为Calendar对象

                   Calendar c7 = Calendar.getInstance();

                   c7.setTime(d);

         5、Calendar对象和相对时间之间的互转

                            Calendar c8 = Calendar.getInstance();

                            long t = 1252785271098L;

                            //将Calendar对象转换为相对时间

                            long t1 = c8.getTimeInMillis();

                            //将相对时间转换为Calendar对象

                            Calendar c9 = Calendar.getInstance();

                            c9.setTimeInMillis(t1);

在转换时，使用Calendar类中的getTimeInMillis方法可以将Calendar对象转换为相对时间。在将相对时间转换为Calendar对象时，首先创建一个Calendar对象，然后再使用Calendar类的setTimeInMillis方法设置时间即可。

**9.6.1.3 应用示例**

         下面以两个简单的示例介绍时间和日期处理的基本使用。

         1、计算两个日期之间相差的天数

例如计算2010年4月1号和2009年3月11号之间相差的天数，则可以使用时间和日期处理进行计算。

该程序实现的原理为：首先代表两个特定的时间点，这里使用Calendar的对象进行代表，然后将两个时间点转换为对应的相对时间，求两个时间点相对时间的差值，然后除以1天的毫秒数(24小时X60分钟X60秒X1000毫秒)即可获得对应的天数。实现该示例的完整代码如下：

         import java.util.\*;

/\*\*

 \* 计算两个日期之间相差的天数

 \*/

public class DateExample1 {

         public static void main(String[] args) {

                   //设置两个日期

                   //日期：2009年3月11号

                   Calendar c1 = Calendar.getInstance();

                   c1.set(2009, 3 - 1, 11);

                   //日期：2010年4月1号

                   Calendar c2 = Calendar.getInstance();

                   c2.set(2010, 4 - 1, 1);

                   //转换为相对时间

                   long t1 = c1.getTimeInMillis();

                   long t2 = c2.getTimeInMillis();

                   //计算天数

                   long days = (t2 - t1)/(24 \* 60 \* 60 \* 1000);

                   System.out.println(days);

         }

}

         2、输出当前月的月历

该示例的功能是输出当前系统时间所在月的日历，例如当前系统时间是2009年3月10日，则输出2009年3月的日历。

该程序实现的原理为：首先获得该月1号是星期几，然后获得该月的天数，最后使用流程控制实现按照日历的格式进行输出即可。即如果1号是星期一，则打印一个单位的空格，如果1号是星期二，则打印两个单位的空格，依次类推。打印完星期六的日期以后，进行换行。实现该示例的完整代码如下：

         import java.util.\*;

/\*\*

 \* 输出当前月的日历

 \*/

public class DateExample2{

         public static void main(String[] args){

                   //获得当前时间

                   Calendar c = Calendar.getInstance();

                   //设置代表的日期为1号

                   c.set(Calendar.DATE,1);

                   //获得1号是星期几

                   int start = c.get(Calendar.DAY\_OF\_WEEK);

                   //获得当前月的最大日期数

                   int maxDay = c.getActualMaximum(Calendar.DATE);

                   //输出标题

                   System.out.println("星期日 星期一 星期二 星期三  星期四  星期五   星期六");

                   //输出开始的空格

                   for(int i = 1;i < start;i++){

                            System.out.print("      ");

                   }

                   //输出该月中的所有日期

                   for(int i = 1;i <= maxDay;i++){

                            //输出日期数字

                            System.out.print(" " + i);

                            //输出分隔空格

                            System.out.print("     ");

                            if(i < 10){

                                     System.out.print(' ');

                            }

                            //判断是否换行

                            if((start + i - 1) % 7 == 0){

                                     System.out.println();

                            }

                   }

                   //换行

                   System.out.println();

         }

}

         关于时间和日期的处理就介绍这么多，更多的实现方法还需要根据具体问题进行对应的实现。

**分享到：** 