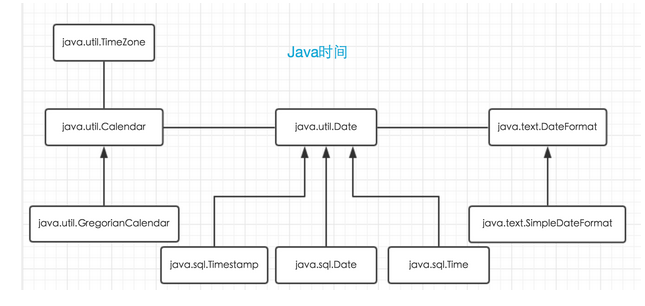
Java核心API(日期时间处理)



# java时间API

## Date

java.util.Date对象表示一个精确到毫秒的瞬间; 但由于Date从JDK1.0起就开始存在了,历史悠久,而且功能强大(既包含日期,也包含时间),所以他的大部分构造器/方法都已Deprecated,因此就不再推荐使用(如果贸然使用的话,可能会出现性能/安全方面的问题);下面我仅介绍它还剩下的为数不多的几个方法(这些方法的共同点是Date与毫秒值的转换):

构造器：

* Date(): 在底层调用System.currentTimeMillis()作为日期参数.
* Date(long date): 根据指定的long整数(从1970-1-1 00:00:00以来经过的毫秒数)来生成Date对象.

方法

* boolean after(Date when): 测试this日期是否在指定日期when之后;
* boolean before(Date when): 测试this日期是否在指定日期when之前;
* long getTime(): 获取从1979-01-01 00:00:00 到Date对象之间经过的毫秒值;
* void setTime(long time): 设置时间,time含义上同.

|  |
| --- |
| @Test  public void test10() {  Date dateBefore = new Date();  Date dateAfter = new Date(System.currentTimeMillis() + 1);  System.out.println("before: " + dateBefore.getTime());  System.out.println("after: " + dateAfter.getTime());  System.out.println(dateBefore.before(dateAfter));  System.out.println(dateAfter.after(dateBefore));  dateBefore.setTime(System.currentTimeMillis());  System.out.println(dateBefore.getTime());  System.out.println(dateBefore.before(dateAfter));  }  //before: 1473303995409  //after: 1473303995410  //true  //true  //1473303995410  //false |

## Calendar

Calendar类： java.util.Calendar类用于封装日历信息，其主作用在于其方法可以对时间分量进行运算。Calendar类提供了大量访问/修改日期/时间的方法, 常用的方法如下:

| **Method** | **Description** |
| --- | --- |
| void add(int field, int amount) | Adds or subtracts the specified amount of time to the given calendar field, based on the calendar’s rules. |
| int get(int field) | Returns the value of the given calendar field. |
| int getActualMaximum(int field) | Returns the maximum value that the specified calendar field could have, given the time value of this Calendar. |
| int getActualMinimum(int field) | Returns the minimum value that the specified calendar field could have, given the time value of this Calendar. |
| void roll(int field, int amount) | Adds the specified (signed) amount to the specified calendar field without changing larger fields. |
| void set(int field, int value) | Sets the given calendar field to the given value. |
| void set(int year, int month, int date) | Sets the values for the calendar fields YEAR, MONTH, and DAY\_OF\_MONTH. |
| void set(int year, int month, int date, int hourOfDay, int minute, int second) | Sets the values for the fields YEAR, MONTH, DAY\_OF\_MONTH, HOUR, MINUTE, and SECOND. |
| void setTimeInMillis(long millis) | Sets this Calendar’s current time from the given long value. |
| long getTimeInMillis() | Returns this Calendar’s time value in milliseconds. |
| int类型的field参数, field是Calendar类的类变量,：  Calendar.YEAR  Calendar.MONTH （从0开始，代表1月）  Calendar.DATE  Calendar.DAY\_OF\_WEEK代表的星期, 起始值是周日  Calendar.DAY\_OF\_MONTH 月里边的天---几号  Calendar.DAY\_OF\_WEEK 星期里的天---星期几  Calendar.DAY\_OF\_YEAR 年里的天  Calendar.HOUR\_OF\_DAY , Calendar.MINUTE, Calendar.SECOND | |

1）通过Calendar的静态方法获取一个实例该方法会根据当前系统所在地区来自行决定时区，帮我们创建Calendar实例，这里要注意，实际上根据不同的地区，Calendar有若干个子类实现。而Calendar本身是抽象类，不能被实例化！我们不需要关心创建的具体实例为哪个子类，我们只需要根据Calendar规定的方法来使用就可以了。使用Calendar.getInstance();获取的就是默认的GregorianCalendar,getInstance()方法的内部会调用cal = new GregorianCalendar(zone, aLocale);

2）日历类所解决的根本问题是简化日期的计算，要想表示某个日期还应该使用Date类描述。Calendar是可以将其描述的时间转化为Date的，我们只需要调用其getTime()方法就可以获取描述的日期的Date对象了。

|  |
| --- |
| @Test  public void test16() {  Calendar calendar = Calendar.getInstance();  Date date = calendar.getTime();  Calendar newCalendar = Calendar.getInstance();  newCalendar.setTime(date);  System.out.println(calendar.get(Calendar.DATE));  } |

3) 设置日期

|  |
| --- |
| Calendar calendar=Calendar.getInstance();// 构造出来表示当前时间的日历类  Date now=calendar.getTime();// 获取日历所描述的日期  calendar.set(Calendar.YEAR, 2012);// 设置日历表示2012年  calendar.set(Calendar.DAY\_OF\_MONTH,15);// 设置日历表示15号  calendar.add(Calendar.DAY\_OF\_YEAR, 22);// 想得到22天以后是哪天  calendar.add(Calendar.DAY\_OF\_YEAR, -5);//5天以前是哪天  calendar.add(Calendar.MONTH, 1); // 得到1个月后是哪天  System.out.println(calendar.getTime());// 获取日历所描述的 |

4）获取当前日历表示的日期中的某个时间单位可以使用get方法.

|  |
| --- |
| int year=calendar.get(Calendar.YEAR);  int month=calendar.get(Calendar.MONTH);  int day=calendar.get(Calendar.DAY\_OF\_MONTH);  System.out.println(year+"年"+(month+1)+"月"+day+"日");//month要处理由于Date存在缺陷,所以JDK又提供了java.util.Calendar来处理日期和时间.Calendar是一个抽象类,是所有日历类的模板,因此,我们可以继承Calendar来实现其他的历法(比如阴历); |

## DateFormat

java.text.DateFormat是一个抽象类, 他提供了如下几个方法获取DateFormat对象.

| **方法** | **描述** |
| --- | --- |
| static DateFormat getDateInstance() | Gets the date formatter with the default formatting style for the default locale. |
| static DateFormat getDateTimeInstance() | Gets the date/time formatter with the default formatting style for the default locale. |
| static DateFormat getTimeInstance() | Gets the time formatter with the default formatting style for the default locale. |

其实上面三个方法还可以指定日期/时间的样式, 如FULL/LONG/MEDIUM/SHOT, 通过这四个样式参数可以控制生成的格式化字符串. 但由于在我们的实际开发中很少直接用DateFormat类,因此就不对其做过多的介绍.而我们比较常用的是其子类SimpleDateFormat(其实上面几个getXxxInstance方法返回的也是SimpleDateFormat实例)

## SimpleDateFormat

java.text.SimpleDateFormat可以非常灵活的格式化Date, 也可以用于解析各种格式的日期字符串.创建SimpleDateFormat对象时需要传入一个pattern字符串,这个pattern不是正则表达式,而是一个日期模板字符串.

|  |
| --- |
| @Test  public void test17() {  try {  DateFormat format = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");  Date date = new Date();  // Date转化为String  System.out.println(format.format(date));  String timeString = "2015-12-30 08:53:21";  // String转化为Date  Date newDate;  newDate = format.parse(timeString);  System.out.println(newDate);  } catch (ParseException e) {  e.printStackTrace();  }  } |

在时间日期格式化时, 有下面几个方法是最常用的:

| **方法** | **描述** | **小结** |
| --- | --- | --- |
| String format(Date date) | Formats a Date into a date/time string. | Date -> String |
| Date parse(String source) | Parses text from the beginning of the given string to produce a date. | String -> Date |

# 配置单例Formatter

由于在一个项目中时间格式化和解析的格式一般只有一种(我们应该不希望格式化之后的time到最后反而解析不出来),因此我们没有必要每次使用时都new出一个Formatter来,这样不光会造成性能下降还有可能造成时间形式不统一而出错.因此,我们可以在Spring的容器中装载一个Formatter Bean,使用时@Autowired就可以了:

|  |
| --- |
| <!-- 配置时间格式化器 -->  <bean id="dateFormatter" class="java.text.SimpleDateFormat">  <constructor-arg value="yyyy-MM-dd HH:mm:ss"/>  </bean> |
| @Autowired  private DateFormat dateFormatter;  ...  String time = dateFormatter.format(System.currentTimeMillis()); |