ImportNew

Java8 日期/时间（Date Time）API指南

2014/12/22 | 分类： 基础技术 | 0 条评论 | 标签： date, java8, time

35 本文由 ImportNew - Justin Wu 翻译自 journaldev。欢迎加入翻译小组。转载请见文末要求。

Java 8日期/时间（ Date/Time ）API是开发人员最受追捧的变化之一，Java从一开始就没有对日期时间处理的一致性方法，因此日期/时间API也是除Java核心API以外另一项倍受欢迎的内容。

为什么我们需要新的Java日期/时间API？

在开始研究Java 8日期/时间API之前，让我们先来看一下为什么我们需要这样一个新的API。在Java中，现有的与日期和时间相关的类存在诸多问题，其中有：

1. Java的日期/时间类的定义并不一致，在java.util和java.sql的包中都有日期类，此外用于格式化和解析的类在java.text包中定义。
2. java.util.Date同时包含日期和时间，而java.sql.Date仅包含日期，将其纳入java.sql包并不合理。另外这两个类都有相同的名字，这本身就是一个非常糟糕的设计。
3. 对于时间、时间戳、格式化以及解析，并没有一些明确定义的类。对于格式化和解析的需求，我们有java.text.DateFormat抽象类，但通常情况下，SimpleDateFormat类被用于此类需求。
4. 所有的日期类都是可变的，因此他们都不是线程安全的，这是Java日期类最大的问题之一。
5. 日期类并不提供国际化，没有时区支持，因此Java引入了java.util.Calendar和java.util.TimeZone类，但他们同样存在上述所有的问题。

在现有的日期和日历类中定义的方法还存在一些其他的问题，但以上问题已经很清晰地表明：Java需要一个健壮的日期/时间类。这也是为什么Joda Time在Java日期/时间需求中扮演了高质量替换的重要角色。

Java 8日期/时间API

Java 8日期/时间API是JSR-310的实现，它的实现目标是克服旧的日期时间实现中所有的缺陷，新的日期/时间API的一些设计原则是：

1. 不变性：新的日期/时间API中，所有的类都是不可变的，这对多线程环境有好处。
2. 关注点分离：新的API将人可读的日期时间和机器时间（unix timestamp）明确分离，它为日期（Date）、时间（Time）、日期时间（DateTime）、时间戳（unix timestamp）以及时区定义了不同的类。
3. 清晰：在所有的类中，方法都被明确定义用以完成相同的行为。举个例子，要拿到当前实例我们可以使用now()方法，在所有的类中都定义了format()和parse()方法，而不是像以前那样专门有一个独立的类。为了更好的处理问题，所有的类都使用了工厂模式和策略模式，一旦你使用了其中某个类的方法，与其他类协同工作并不困难。
4. 实用操作：所有新的日期/时间API类都实现了一系列方法用以完成通用的任务，如：加、减、格式化、解析、从日期/时间中提取单独部分，等等。
5. 可扩展性：新的日期/时间API是工作在ISO-8601日历系统上的，但我们也可以将其应用在非ISO的日历上。

Java日期/时间API包

Java日期/时间API包含以下相应的包。

1. java.time包：这是新的Java日期/时间API的基础包，所有的主要基础类都是这个包的一部分，如：LocalDate, LocalTime, LocalDateTime, Instant, Period, Duration等等。所有这些类都是不可变的和线程安全的，在绝大多数情况下，这些类能够有效地处理一些公共的需求。
2. java.time.chrono包：这个包为非ISO的日历系统定义了一些泛化的API，我们可以扩展AbstractChronology类来创建自己的日历系统。
3. java.time.format包：这个包包含能够格式化和解析日期时间对象的类，在绝大多数情况下，我们不应该直接使用它们，因为java.time包中相应的类已经提供了格式化和解析的方法。
4. java.time.temporal包：这个包包含一些时态对象，我们可以用其找出关于日期/时间对象的某个特定日期或时间，比如说，可以找到某月的第一天或最后一天。你可以非常容易地认出这些方法，因为它们都具有“withXXX”的格式。
5. java.time.zone包：这个包包含支持不同时区以及相关规则的类。

Java日期/时间API示例

我们已经浏览了Java日期/时间API的大多数重要部分，现在是时候根据示例仔细看一下最重要的一些类了。
1. java.time.LocalDate：LocalDate是一个不可变的类，它表示默认格式(yyyy-MM-dd)的日期，我们可以使用now()方法得到当前时间，也可以提供输入年份、月份和日期的输入参数来创建一个LocalDate实例。该类为now()方法提供了重载方法，我们可以传入ZoneId来获得指定时区的日期。该类提供与java.sql.Date相同的功能，对于如何使用该类，我们来看一个简单的例子。

```
package com.journaldev.java8.time;

import java.time.LocalDate;
import java.time.Month;
import java.time.ZoneId;

/**
 * LocalDate Examples
 * @author pankaj
 *
 */
public class LocalDateExample {

    public static void main(String[] args) {

        //Current Date
        LocalDate today = LocalDate.now();
        System.out.println("Current Date="+today);

        //Creating LocalDate by providing input arguments
        LocalDate firstDay_2014 = LocalDate.of(2014, Month.JANUARY, 1);
        System.out.println("Specific Date="+firstDay_2014);

        //Try creating date by providing invalid inputs
        //LocalDate feb29_2014 = LocalDate.of(2014, Month.FEBRUARY, 29);
        //Exception in thread "main" java.time.DateTimeException:
        //Invalid date 'February 29' as '2014' is not a leap year

        //Current date in "Asia/Kolkata", you can get it from ZoneId javadoc
        LocalDate todayKolkata = LocalDate.now(ZoneId.of("Asia/Kolkata"));
        System.out.println("Current Date in IST="+todayKolkata);

        //java.time.zone.ZoneRulesException: Unknown time-zone ID: IST
        //LocalDate todayIST = LocalDate.now(ZoneId.of("IST"));

        //Getting date from the base date i.e 01/01/1970
        LocalDate dateFromBase = LocalDate.ofEpochDay(365);
        System.out.println("365th day from base date= "+dateFromBase);

        LocalDate hundredDay2014 = LocalDate.ofYearDay(2014, 100);
        System.out.println("100th day of 2014="+hundredDay2014);
    }

}
```

示例方法的详解都包含在注释内，当我们运行程序时，可以得到以下输出：

```
Current Date=2014-04-28
Specific Date=2014-01-01
Current Date in IST=2014-04-29
365th day from base date= 1971-01-01
100th day of 2014=2014-04-10
```

2. java.time.LocalTime：LocalTime是一个不可变的类，它的实例代表一个符合人类可读格式的时间，默认格式是hh:mm:ss.zzz。像LocalDate一样，该类也提供了时区支持，同时也可以传入小时、分钟和秒等输入参数创建实例，我们来看一个简单的程序，演示该类的使用方法。

```
package com.journaldev.java8.time;

import java.time.LocalTime;
import java.time.ZoneId;

```

```
/**
 * LocalTime Examples
 * @author pankaj
 *
 */
public class LocalTimeExample {

    public static void main(String[] args) {

        //Current Time
        LocalTime time = LocalTime.now();
        System.out.println("Current Time="+time);

        //Creating LocalTime by providing input arguments
        LocalTime specificTime = LocalTime.of(12,20,25,40);
        System.out.println("Specific Time of Day="+specificTime);

        //Try creating time by providing invalid inputs
        //LocalTime invalidTime = LocalTime.of(25,20);
        //Exception in thread "main" java.time.DateTimeException:
        //Invalid value for HourOfDay (valid values 0 - 23): 25

        //Current date in "Asia/Kolkata", you can get it from ZoneId javadoc
        LocalTime timeKolkata = LocalTime.now(ZoneId.of("Asia/Kolkata"));
        System.out.println("Current Time in IST="+timeKolkata);

        //java.time.zone.ZoneRulesException: Unknown time-zone ID: IST
        //LocalTime todayIST = LocalTime.now(ZoneId.of("IST"));

        //Getting date from the base date i.e 01/01/1970
        LocalTime specificSecondTime = LocalTime.ofSecondOfDay(10000);
        System.out.println("10000th second time= "+specificSecondTime);

    }
}
```

当运行以上程序时，可以看到如下输出。

```
Current Time=15:51:45.240
Specific Time of Day=12:20:25.000000040
Current Time in IST=04:21:45.276
10000th second time= 02:46:40
```

3. java.time.LocalDateTime：LocalDateTime是一个不可变的日期-时间对象，它表示一组日期-时间，默认格式是yyyy-MM-dd-HH-mm-ss.zzz。它提供了一个工厂方法，接收LocalDate和LocalTime输入参数，创建LocalDateTime实例。我们来看一个简单的例子。

```
package com.journaldev.java8.time;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.Month;
import java.time.ZoneId;
import java.time.ZoneOffset;

public class LocalDateTimeExample {

    public static void main(String[] args) {

        //Current Date
        LocalDateTime today = LocalDateTime.now();
        System.out.println("Current DateTime="+today);

        //Current Date using LocalDate and LocalTime
        today = LocalDateTime.of(LocalDate.now(), LocalTime.now());
        System.out.println("Current DateTime="+today);

        //Creating LocalDateTime by providing input arguments
        LocalDateTime specificDate = LocalDateTime.of(2014, Month.JANUARY, 1, 10, 10, 30);
        System.out.println("Specific Date="+specificDate);

        //Try creating date by providing invalid inputs
        //LocalDateTime feb29_2014 = LocalDateTime.of(2014, Month.FEBRUARY, 28, 25,1,1);
        //Exception in thread "main" java.time.DateTimeException:
        //Invalid value for HourOfDay (valid values 0 - 23): 25

        //Current date in "Asia/Kolkata", you can get it from ZoneId javadoc
        LocalDateTime todayKolkata = LocalDateTime.now(ZoneId.of("Asia/Kolkata"));
        System.out.println("Current Date in IST="+todayKolkata);

        //java.time.zone.ZoneRulesException: Unknown time-zone ID: IST
        //LocalDateTime todayIST = LocalDateTime.now(ZoneId.of("IST"));

        //Getting date from the base date i.e 01/01/1970
        LocalDateTime dateFromBase = LocalDateTime.ofEpochSecond(10000, 0, ZoneOffset.UTC);
        System.out.println("10000th second time from 01/01/1970= "+dateFromBase);

    }
}
```

在所有这三个例子中，我们已经看到如果我们提供了无效的参数去创建日期/时间，那么系统会抛出java.time.DateTimeException，这是一种运行时异常，我们并不需要显式地捕获它。
同时我们也看到，能够通过传入ZoneId得到日期/时间数据，你可以从它的Javadoc中得到支持的Zoneid的列表，当运行以上类时，可以得到以下输出。

```
Current DateTime=2014-04-28T16:00:49.455
Current DateTime=2014-04-28T16:00:49.493
Specific Date=2014-01-01T10:10:30
Current Date in IST=2014-04-29T04:30:49.493
10000th second time from 01/01/1970= 1970-01-01T02:46:40
```

4. java.time.Instant：Instant类是用在机器可读的时间格式上的，它以Unix时间戳的形式存储日期时间，我们来看一个简单的程序。

```
package com.journaldev.java8.time;

import java.time.Duration;
import java.time.Instant;

public class InstantExample {

    public static void main(String[] args) {
        //Current timestamp
        Instant timestamp = Instant.now();
        System.out.println("Current Timestamp = "+timestamp);

        //Instant from timestamp
        Instant specificTime = Instant.ofEpochMilli(timestamp.toEpochMilli());
        System.out.println("Specific Time = "+specificTime);

        //Duration example
        Duration thirtyDay = Duration.ofDays(30);
        System.out.println(thirtyDay);
    }

}
```

上述程序的输出是：

```
Current Timestamp = 2014-04-28T23:20:08.489Z
Specific Time = 2014-04-28T23:20:08.489Z
PT720H
```

5. 日期API工具：我们早些时候提到过，大多数日期/时间API类都实现了一系列工具方法，如：加/减天数、周数、月份数，等等。还有其他的工具方法能够使用TemporalAdjuster调整日期，并计算两个日期间的周期。

```
package com.journaldev.java8.time;

import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.time.temporal.TemporalAdjusters;

public class DateAPIUtilities {

    public static void main(String[] args) {

        LocalDate today = LocalDate.now();

        //Get the Year, check if it's leap year
        System.out.println("Year "+today.getYear()+" is Leap Year? "+today.isLeapYear());

        //Compare two LocalDate for before and after
        System.out.println("Today is before 01/01/2015? "+today.isBefore(LocalDate.of(2015,1,1)));

        //Create LocalDateTime from LocalDate
        System.out.println("Current Time="+today.atTime(LocalTime.now()));

        //plus and minus operations
        System.out.println("10 days after today will be "+today.plusDays(10));
        System.out.println("3 weeks after today will be "+today.plusWeeks(3));
        System.out.println("20 months after today will be "+today.plusMonths(20));

        System.out.println("10 days before today will be "+today.minusDays(10));
        System.out.println("3 weeks before today will be "+today.minusWeeks(3));
        System.out.println("20 months before today will be "+today.minusMonths(20));

        //Temporal adjusters for adjusting the dates
        System.out.println("First date of this month= "+today.with(TemporalAdjusters.firstDayOfMonth()));
        LocalDate lastDayOfYear = today.with(TemporalAdjusters.lastDayOfYear());
        System.out.println("Last date of this year= "+lastDayOfYear);

        Period period = today.until(lastDayOfYear);
        System.out.println("Period Format= "+period);
        System.out.println("Months remaining in the year= "+period.getMonths());
    }
}
```

上述程序的输出是：

```
Year 2014 is Leap Year? false
Today is before 01/01/2015? true
Current Time=2014-04-28T16:23:53.154
10 days after today will be 2014-05-08
3 weeks after today will be 2014-05-19
20 months after today will be 2015-12-28
10 days before today will be 2014-04-18
3 weeks before today will be 2014-04-07
20 months before today will be 2012-08-28
First date of this month= 2014-04-01
Last date of this year= 2014-12-31
Period Format= P8M3D
Months remaining in the year= 8
```

6. 解析和格式化：将一个日期格式转换为不同的格式，之后再解析一个字符串，得到日期时间对象，这些都是很常见的。我们来看一下简单的例子。

```
package com.journaldev.java8.time;

import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class DateParseFormatExample {

    public static void main(String[] args) {

        //Format examples
        LocalDate date = LocalDate.now();
        //default format
        System.out.println("Default format of LocalDate="+date);
        //specific format
        System.out.println(date.format(DateTimeFormatter.ofPattern("d::MMM::uuuu")));
        System.out.println(date.format(DateTimeFormatter.BASIC_ISO_DATE));

        LocalDateTime dateTime = LocalDateTime.now();
        //default format
        System.out.println("Default format of LocalDateTime="+dateTime);
        //specific format
        System.out.println(dateTime.format(DateTimeFormatter.ofPattern("d::MMM::uuuu HH::mm::ss")));
        System.out.println(dateTime.format(DateTimeFormatter.BASIC_ISO_DATE));

        Instant timestamp = Instant.now();
        //default format
        System.out.println("Default format of Instant="+timestamp);

        //Parse examples
        LocalDateTime dt = LocalDateTime.parse("27::Apr::2014 21::39::48",
                DateTimeFormatter.ofPattern("d::MMM::uuuu HH::mm::ss"));
        System.out.println("Default format after parsing = "+dt);
    }

}
```

当运行以上程序时，可以看到如下输出。

```
Default format of LocalDate=2014-04-28
28::Apr::2014
20140428
Default format of LocalDateTime=2014-04-28T16:25:49.341
28::Apr::2014 16::25::49
20140428
Default format of Instant=2014-04-28T23:25:49.342Z
Default format after parsing = 2014-04-27T21:39:48
```

7. 旧的日期时间支持：旧的日期/时间类已经在几乎所有的应用程序中使用，因此做到向下兼容是必须的。这也是为什么会有若干工具方法帮助我们将旧的类转换为新的类，反之亦然。我们来看一下简单的例子。

```
package com.journaldev.java8.time;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.TimeZone;

public class DateAPILegacySupport {

    public static void main(String[] args) {

        //Date to Instant
        Instant timestamp = new Date().toInstant();
        //Now we can convert Instant to LocalDateTime or other similar classes
        LocalDateTime date = LocalDateTime.ofInstant(timestamp,
                        ZoneId.of(ZoneId.SHORT_IDS.get("PST")));
        System.out.println("Date = "+date);

        //Calendar to Instant
        Instant time = Calendar.getInstance().toInstant();
        System.out.println(time);
        //TimeZone to ZoneId
        ZoneId defaultZone = TimeZone.getDefault().toZoneId();
        System.out.println(defaultZone);

        //ZonedDateTime from specific Calendar
        ZonedDateTime gregorianCalendarDateTime = new GregorianCalendar().toZonedDateTime();
        System.out.println(gregorianCalendarDateTime);

        //Date API to Legacy classes
        Date dt = Date.from(Instant.now());
        System.out.println(dt);

        TimeZone tz = TimeZone.getTimeZone(defaultZone);
        System.out.println(tz);

        GregorianCalendar gc = GregorianCalendar.from(gregorianCalendarDateTime);
        System.out.println(gc);
```

```
43
44      }
45
46  }
```

当运行以上程序时，可以看到如下输出。

```
Date = 2014-04-28T16:28:54.340
2014-04-28T23:28:54.395Z
America/Los_Angeles
2014-04-28T16:28:54.404-07:00[America/Los_Angeles]
Mon Apr 28 16:28:54 PDT 2014
sun.util.calendar.ZoneInfo[id="America/Los_Angeles",offset=-28800000,dstSavings=3600000,useDaylight=true,transitions=185,lastRule=java.util.Si
java.util.GregorianCalendar[time=1398727734404,areFieldsSet=true,areAllFieldsSet=true,lenient=true,zone=sun.util.calendar.ZoneInfo[id="America
```

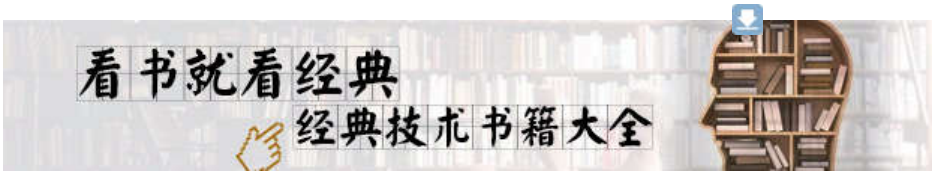
你可以看到，旧的TimeZone和GregorianCalendar类的toString()方法太啰嗦了，一点都不友好。
这就是所有的Java 8 日期/时间API的内容，我非常喜欢这个API，它易于使用，同时它采取了某项工作，使相似的方法也易于寻找，虽然从旧的类转移到新的日期时间类需要消耗一定的时间，但我相信这是值得的。
原文链接： journaldev 翻译： ImportNew.com - Justin Wu
译文链接： http://www.importnew.com/14140.html
[**转载请保留原文出处、译者和译文链接。**]

关于作者： Justin Wu

（新浪微博：@微风12345678）

查看Justin Wu的更多文章 >>

35



相关文章

- Java8学习笔记
- Java 8开发的4大顶级技巧
- 使用 Java8 Optional 的正确姿势
- Java 8 –行为参数化
- Java8 Top Tips
- Java8中CAS的增强
- Java 8 指南
- Java Date Time 教程-java.util.Date
- Java8系列之重新认识HashMap
- Java8初体验（2）：Stream语法详解

发表评论

Comment form

Name*

姓名

邮箱*

请填写邮箱

网站 (请以 http://开头)

请填写网站地址

评论内容*

请填写评论内容

(*) 表示必填项

提交评论

还没有评论。

« SSH框架总结（框架分析+环境搭建+实例源码下载）
Spring实战：为测试方法重置自增列 »

Search for:

Search

Search

- 本周热门文章
- 本月热门
- 热门标签

0 Quartz Cron Expressions 详解

1 使用 Eclipse Checkstyle Plugin

2 RabbitMQ指南（下）

3 OAuth 2.0 认证的原理与实践

4 从Java进程里dump出类的class文件的小工具-...

5 使用 PMD Eclipse插件

6 为什么我们迫切需要持续集成（Continuous I...

7 使用 TeamCity 实现持续集成（C...

8 RESTful API 设计最佳实践

9 Java基础中一些值得聊的话题（加载...

最新评论

- Re: Quartz Cron Expressions ...
  有的，可以参见文章末尾的链接。原文传送“The ‘W’ is used to specify the... 唐尤华
- Re: Quartz Cron Expressions ...
  W解释有误，wiki里说的是最接近的工作日，还有C是不是打错了？没这个字符的吧？ 小杨
- Re: Java 8 Optional类深度解析
  感谢大神分享 九千鸦
- Re: Java面试题全集（上）
  40、怎样将GB2312编码的字符串转换为ISO-8859-1编码的字符串？答：代码如下所示：12S... 代飞
- Re: Java核心技术点之动态代理
  上面自定义的InvocationHandler接口，少继承了java.lang.reflect.In... 张胜
- Re: 跟我一起学Spring 3(4)–...
  写的不错！！！楼主好样的，解惑啦！ sunshine
- Re: 记一次synchronized锁字符串引发的...
  请问JdkUtil是一个什么类？ wyh66
- Re: 类在什么时候加载和初始化
  学习了 zhangliao613

关于ImportNew

ImportNew 专注于 Java 技术分享。于2012年11月11日 11:11正式上线。是的，这是一个很特别的时刻 :)

ImportNew 由两个 Java 关键字 import 和 new 组成，意指：Java 开发者学习新知识的网站。 import 可认为是学习和吸收， new 则可认为是新知识、新技术圈子和新朋友……

联系我们

Email：ImportNew.com@gmail.com
新浪微博：@ImportNew
推荐微信号

推荐关注