Java 核心技术

第八章 Java常用类

第四节 时间相关类

华东师范大学 陈良育

时间类(1)

- java.util.Date(基本废弃，Deprecated)
  - getTime()，返回自1970.1.1以来的毫秒数
- java.sql.Date（和数据库对应的时间类）
- Calendar是目前程序中最常用的，但是是抽象类
  - Calendar gc=Calendar.getInstance();
  - Calendar gc= new GregorianCalendar();
  - 简单工厂模式
  - 查看CalendarClassTest.java

时间类(2)

- Calendar
  - get(Field) 来获取时间中每个属性的值. 注意，月份0-11.
  - getTime()，返回相应的Date对象
  - getTimeInMillis(), 返回自1970.1.1以来的毫秒数
  - set(Field) 设置时间字段
  - add(field, amount) 根据指定字段增加/减少时间
  - roll(field, amount) 根据指定字段增加/减少时间，但不影响上一级的时间段

时间类(3)

- Java 8 推出新的时间API
  - java.time包
  - 旧的设计不好（重名的类、线程不安全等）
  - 新版本优点
    - 不变性，在多线程环境下
    - 遵循设计模式，设计得更好，可扩展性强

时间类(4)

- Java 8 时间包概述
  - java.time包：新的Java日期/时间API的基础包
  - java.time.chrono包：为非ISO的日历系统定义了一些泛化的API，
  - java.time.format包：格式化和解析日期时间对象的类
  - java.time.temporal包：包含一些时态对象，可以用其找出关于日期/时间对象的某个特定日期或时间
  - java.time.zone包：包含支持不同时区以及相关规则的类

时间类(5)

- Java 8 java.time包主要类
  - LocalDate： 日期类
  - LocalTime： 时间类(时分秒-纳秒)
  - LocalDateTime: LocalDate + LocalTime
  - Instant: 时间戳
- 查看java.time的示例代码

总结

- 当前多数程序还是使用Calendar类处理时间
- 学习Java 8 的时间新特性并应用

代码(1) DateTest.java


```java
import java.util.Date;

public class DateTest {

    public static void main(String[] args) {
        Date d = new Date();
        System.out.println(d);
        System.out.println(d.getTime());
        //the number of milliseconds since 1970.1.1 00:00:00

    }

}
```

代码(2) CalendarClassTest.java


```java
import java.util.Calendar;

public class CalendarClassTest {

    public static void main(String[] args) {
        Calendar gc = Calendar.getInstance();
        System.out.println(gc.getClass().getName());
        //Calendar.getInstance();返回的是GregorianCalendar对象

        GregorianCalendar gc2 = new GregorianCalendar();
        System.out.println(gc2.getClass().getName());
    }

}
```

代码(3) CalendarTest.java


```java
import java.util.Calendar;

public class CalendarTest {

    Calendar calendar = Calendar.getInstance();

    public void test1() {
        // 获取年
        int year = calendar.get(Calendar.YEAR);
        // 获取月，这里需要需要月份的范围为0~11，因此获取月份的时候需要+1才是当前月份值
        int month = calendar.get(Calendar.MONTH) + 1;
        // 获取日
        int day = calendar.get(Calendar.DAY_OF_MONTH);

        // 获取时
        int hour = calendar.get(Calendar.HOUR);
        // int hour = calendar.get(Calendar.HOUR_OF_DAY); // 24小时表示
        // 获取分
        int minute = calendar.get(Calendar.MINUTE);
        // 获取秒
        int second = calendar.get(Calendar.SECOND);

        // 星期，英语国家星期从星期日开始计算
        int weekday = calendar.get(Calendar.DAY_OF_WEEK);

        System.out.println("现在是" + year + "年" + month + "月" + day + "日" + hour
                + "时" + minute + "分" + second + "秒" + "星期" + weekday);
    }
```

代码(4) CalendarTest.java


```java
// 一年后的今天
public void test2() {
    // 同理换成下个月的今天calendar.add(Calendar.MONTH, 1);
    calendar.add(Calendar.YEAR, 1);

    // 获取年
    int year = calendar.get(Calendar.YEAR);
    // 获取月
    int month = calendar.get(Calendar.MONTH) + 1;
    // 获取日
    int day = calendar.get(Calendar.DAY_OF_MONTH);

    System.out.println("一年后的今天：" + year + "年" + month + "月" + day + "日");
}

// 获取任意一个月的最后一天
public void test3() {
    // 假设求6月的最后一天
    int currentMonth = 6;
    // 先求出7月份的第一天，实际中这里6为外部传递进来的currentMonth变量
    // 1
    calendar.set(calendar.get(Calendar.YEAR), currentMonth, 1);

    calendar.add(Calendar.DATE, -1);

    // 获取日
    int day = calendar.get(Calendar.DAY_OF_MONTH);

    System.out.println("6月份的最后一天为" + day + "号");
}
```

代码(5) CalendarTest.java


```java
// 设置日期
public void test4() {
    calendar.set(Calendar.YEAR, 2000);
    System.out.println("现在是" + calendar.get(Calendar.YEAR) + "年");

    calendar.set(2018, 7, 8);
    // 获取年
    int year = calendar.get(Calendar.YEAR);
    // 获取月
    int month = calendar.get(Calendar.MONTH)+1;
    // 获取日
    int day = calendar.get(Calendar.DAY_OF_MONTH);

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

代码(6) CalendarTest.java


```java
//add和roll的区别
public void test5() {

    calendar.set(2018, 7, 8);
    calendar.add(Calendar.DAY_OF_MONTH, -8);

    // 获取年
    int year = calendar.get(Calendar.YEAR);
    // 获取月
    int month = calendar.get(Calendar.MONTH)+1;
    // 获取日
    int day = calendar.get(Calendar.DAY_OF_MONTH);

    System.out.println("2018.8.8, 用add减少8天，现在是" + year + "." + month + "." + day);

    calendar.set(2018, 7, 8);
    calendar.roll(Calendar.DAY_OF_MONTH, -8);

    // 获取年
    year = calendar.get(Calendar.YEAR);
    // 获取月
    month = calendar.get(Calendar.MONTH)+1;
    // 获取日
    day = calendar.get(Calendar.DAY_OF_MONTH);

    System.out.println("2018.8.8, 用roll减少8天，现在是" + year + "." + month + "." + day);
}
```

代码(7) CalendarTest.java


```
    public static void main(String[] args) {
        CalendarTest c = new CalendarTest();
        c.test1();
        System.out.println("============");
        c.test2();
        System.out.println("============");
        c.test3();
        System.out.println("============");
        c.test4();
        System.out.println("============");
        c.test5();

    }

}
```

代码(8) LocalDatexExample.java


```java
import java.time.LocalDate;

public class LocalDateExample {

    public static void main(String[] args) {

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

        //根据指定时间创建LocalDate
        LocalDate firstDay_2014 = LocalDate.of(2014, Month.JANUARY, 1);
        System.out.println("Specific Date="+firstDay_2014);

        //给定错误时间参数，将报异常java.time.DateTimeException
        //LocalDate feb29_2014 = LocalDate.of(2014, Month.FEBRUARY, 29);

        //可以更改时区
        LocalDate todayBeijing = LocalDate.now(ZoneId.of("Asia/Shanghai"));
        System.out.println("Current Date in Shanghai="+todayBeijing);

        //从纪元日01/01/1970开始365天
        LocalDate dateFromBase = LocalDate.ofEpochDay(365);
        System.out.println("365th day from base date= "+dateFromBase);

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

}
```

代码(9) LocalTimeExample.java


```java
import java.time.LocalTime;

public class LocalTimeExample {

    public static void main(String[] args) {

        //当前时间  时分秒 纳秒
        LocalTime time = LocalTime.now();
        System.out.println("Current Time="+time);

        //根据时分秒
        LocalTime specificTime = LocalTime.of(12,20,25,40);
        System.out.println("Specific Time of Day="+specificTime);

        //错误的时间参数 将报DateTimeException
        //LocalTime invalidTime = LocalTime.of(25,20);

        //上海时间
        LocalTime timeSH = LocalTime.now(ZoneId.of("Asia/Shanghai"));
        System.out.println("Current Time in SH="+timeSH);


        //一天当中第几秒
        LocalTime specificSecondTime = LocalTime.ofSecondOfDay(10000);
        System.out.println("10000th second time= "+specificSecondTime);

    }

}
```

代码(10) LocalDateTimeExample.java


```
import java.time.LocalDate;

public class LocalDateTimeExample {

    public static void main(String[] args) {

        //当前日期 时分秒
        LocalDateTime today = LocalDateTime.now();
        System.out.println("Current DateTime="+today);

        //根据日期，时分秒来创建对象
        today = LocalDateTime.of(LocalDate.now(), LocalTime.now());
        System.out.println("Current DateTime="+today);

        //指定具体时间来创建对象
        LocalDateTime specificDate = LocalDateTime.of(2014, Month.JANUARY, 1, 10, 10, 30);
        System.out.println("Specific Date="+specificDate);

        //如时间不对，将报异常DateTimeException
        //LocalDateTime feb29_2014 = LocalDateTime.of(2014, Month.FEBRUARY, 28, 25,1,1);

        //上海时区
        LocalDateTime todayShanghai = LocalDateTime.now(ZoneId.of("Asia/Shanghai"));
        System.out.println("Current Date in Shanghai="+todayShanghai);


        //从01/01/1970 10000秒
        LocalDateTime dateFromBase = LocalDateTime.ofEpochSecond(10000, 0, ZoneOffset.UTC);
        System.out.println("10000th second time from 01/01/1970= "+dateFromBase);
    }
}
```

代码(11) InstantExample.java


```java
import java.time.Duration;

public class InstantExample {

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

        //从毫秒数来创建时间戳
        Instant specificTime = Instant.ofEpochMilli(timestamp.toEpochMilli());
        System.out.println("Specific Time = "+specificTime);

        Date date = Date.from(timestamp);
        System.out.println("current date = " + date);
    }

}
```

代码(12) DateUtil.java


```java
import java.time.LocalDate;

public class DateUtil {

    public static void main(String[] args) {

        LocalDate today = LocalDate.now();

        //判断是否是闰年
        System.out.println("Year "+today.getYear()+" is Leap Year  "+today.isLeapYear());

        //今天和01/01/2015比较
        System.out.println("Today is before 01/01/2015  "+today.isBefore(LocalDate.of(2015,1,1)));

        //当前时分秒
        System.out.println("Current Time="+today.atTime(LocalTime.now()));

        //加减时间
        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));
```

代码(13) DateUtil.java


```
        //调整时间
        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);

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

谢 谢！