38 日期相关新增规约的启示

更新时间：2020-10-22 18:05:33

> 人生的价值，并不是用时间，而是用深度去衡量的。——列夫·托尔斯泰

1. 前言

《手册》泰山版给出了很多日期时间相关的建议。如：日期格式化时，传入 pattern 中表示年份统一使用小写的 y；在日期格式化中分清楚大写的 M 和小写的 m，大小的 H 和小写的 h 分别代表的意义；不要在程序中写死一年为 365 天，避免在公历闰年时出现日期转换错误或程序逻辑错误；使用枚举值来指代月份。如果使用数字，注意 Date，Calendar 等日期相关类的月份 month 取值在 0-11 之间等。

那么这些建议给我们带来的启示又是什么呢？

2. 启示

2.1 尽量使用官方常量和知名项目的工具类

使用常量、枚举值来指代月份，就避免了因不了解月份的范围为 0 - 11 而导致的 BUG。而且常量和枚举的意义就是将不容易记忆和理解的数字变为容易理解和记忆的单词。

实际编码过程中很多人会通过加、减、乘、除运算来计算时间，很容易出错。

建议大家在平时编码过程中如果 JDK 源码中提供了相关函数则优先使用，否则优先使用知名的三方框架。如常见的工具库：joda-time、 commons-lang3、commons-collection4、 guava 等。

如想计算程序运行的时间差，可以使用 commons-lang3 提供的 `org.apache.commons.lang3.time.StopWatch` 类。

```
@Test
public void testStopWatchSimpleGet() {
    final StopWatch watch = new StopWatch();
    assertEquals(0, watch.getTime());
    assertEquals("00:00:00.000", watch.toString());

    watch.start();
    try {
        Thread.sleep(500);
    } catch (final InterruptedException ex) {
        // ignore
    }
    assertTrue(watch.getTime() < 2000);
}
```

该类提供了重置、暂停、恢复和获取时间差的功能，并且时间差支持根据 TimeUnit 来获取。

如果想构造区间可以使用 guava 的 `com.google.common.collect.Range` 类，在 guava 的源码中给出了非常详尽的单元测试，如：

```
public void testContainsAll() {
  Range<Integer> range = Range.closed(3, 5);
  assertTrue(range.containsAll(asList(3, 3, 4, 5)));
  assertFalse(range.containsAll(asList(3, 3, 4, 5, 6)));

  // We happen to know that natural-order sorted sets use a different code
  // path, so we test that separately
  assertTrue(range.containsAll(ImmutableSortedSet.of(3, 3, 4, 5)));
  assertTrue(range.containsAll(ImmutableSortedSet.of(3)));
  assertTrue(range.containsAll(ImmutableSortedSet.<Integer>of()));
  assertFalse(range.containsAll(ImmutableSortedSet.of(3, 3, 4, 5, 6)));

  assertTrue(Range.openClosed(3, 3).containsAll(Collections.<Integer>emptySet()));
}
```

官方以及优秀的开源项目代码一般都会进行严格的单元测试，经历过实践的检验，而且不断优化和迭代，往往提供的功能更丰富，出现问题的概率较低，封装的层次更高，性能普遍较高，逻辑更严谨。

2.2 多看源码

建议大家在日常开发过程中养成偶尔去翻看源码的习惯，多看看类的注释、常用函数注释和源码，多看看常用类的函数列表等。

JDK 和知名项目的源码注释会给出非常详尽的信息，会给出该类的目的，注意事项，甚至常见的用法，对我们学习和理解有极大的帮助。

如 `java.util.Calendar` 源码中清晰地给出 Calendar.SUNDAY 值为 1 ， Calendar.JANUARY 值为 0 ：

```
/**
 * Value of the {@link #DAY_OF_WEEK} field indicating
 * Sunday.
 */
public final static int SUNDAY = 1;

/**
 * Value of the {@link #DAY_OF_WEEK} field indicating
 * Monday.
 */
public final static int MONDAY = 2;

/**
  * Value of the {@link #MONTH} field indicating the
  * first month of the year in the Gregorian and Julian calendars.
  */
public final static int JANUARY = 0;

/**
  * Value of the {@link #MONTH} field indicating the
  * second month of the year in the Gregorian and Julian calendars.
  */
public final static int FEBRUARY = 1;
```

再如 java.time.LocalTime#of(int, int, int) 的 hour 的范围是 0 到 23，稍微不留神可能就会传入 24。

```
/**
    * Obtains an instance of {@code LocalTime} from an hour, minute and second.
    * <p>
    * This returns a {@code LocalTime} with the specified hour, minute and second.
    * The nanosecond field will be set to zero.
    *
    * @param hour  the hour-of-day to represent, from 0 to 23
    * @param minute  the minute-of-hour to represent, from 0 to 59
    * @param second  the second-of-minute to represent, from 0 to 59
    * @return the local time, not null
    * @throws DateTimeException if the value of any field is out of range
    */
   public static LocalTime of(int hour, int minute, int second) {
       HOUR_OF_DAY.checkValidValue(hour);
       if ((minute | second) == 0) {
           return HOURS[hour];  // for performance
       }
       MINUTE_OF_HOUR.checkValidValue(minute);
       SECOND_OF_MINUTE.checkValidValue(second);
       return new LocalTime(hour, minute, second, 0);
   }
```

很多同学学习技术看了容易忘，看了记不住的一个重要原因是把学习当作纯粹的记忆。

学习某个知识点，比如日期时间 Calendar 的 month 取值问题，总是看书上怎么写，然后记忆下来，却从来不主动去源码里看看。

这样会造成看过的知识点可能会忘记，没看过的知识点不知道，而且很难举一反三，学习效果不太好。

另外正如前面所说，大家看源码的时候一定要 **“先猜想，后验证”**，这样才能印象更加深刻，才能学到东西。

这也是一个非常好的学习方法。当你猜想某个类应该有什么功能，某个函数应该包含哪些步骤，然后再去源码中去印证，发现源码和自己的想法非常相符时，说明自己的想法比较靠谱。如果不符，思考为什么要这么写，这样收获才会更大。

3. 总结

《手册》日期时间章节给我们带来的主要启发是：尽量使用官方常量和知名项目的工具类；多看源码。

希望大家能够重视并实践这些原则，在学习过程中能够自己透过现象看本质，学习更抽象层次的知识。

4、思考和练习

1、你还知道哪些日期时间相关的坑？

欢迎在下方留言评论。

}

37 Java避坑宝典

39 集合相关新增规约的启示