## 玩转时间操作 + 面试题

在 JDK 8 之前，Java 语言为我们提供了两个类用于操作时间，它们分别是：java.util.Date 和 java.util.Calendar，但在 JDK 8 的时候为了解决旧时间操作类的一些缺陷，提供了几个新的类，用于操作时间和日期，它们分别是：LocalTime、LocalDateTime、Instant，都位于 java.time 包下。

时间的操作在我们日常的开发中经常见到，比如，业务数据都要记录创建时间和修改时间，并要把这些时间格式化之后显示到前端页面，再比如我们需要计算业务数据的时间间隔等，都离不开对时间的操作，那如何正确而优雅地使用时间？这就是我们接下来要讨论的话题。

### 时间基础知识科普

#### 格林威治时间

格林威治（又译格林尼治）是英国伦敦南郊原格林威治天文台的所在地，它是世界计算时间和地球经度的起点，国际经度会议 1884 年在美国华盛顿召开，会上通过协议，以经过格林威治天文台的经线为零度经线（即本初子午线），作为地球经度的起点，并以格林威治为“世界时区”的起点。

#### 格林威治时间和北京时间的关系

格林威治时间被定义为世界时间，就是 0 时区，北京是东八区。也就是说格林威治时间的 1 日 0 点，对应到北京的时间就是 1 日 8 点。

#### 时间戳

时间戳是指格林威治时间 1970-01-01 00:00:00（北京时间 1970-01-01 08:00:00）起至现在的总秒数。

### JDK 8 之前的时间操作

#### 1 获取时间

Date date = new Date();  
System.out.println(date);  
Calendar calendar = Calendar.getInstance();  
Date time = calendar.getTime();  
System.out.println(time);

#### 2 获取时间戳

long ts = new Date().getTime();  
System.out.println(ts);  
long ts2 = System.currentTimeMillis();  
System.out.println(ts2);  
long ts3 = Calendar.getInstance().getTimeInMillis();  
System.out.println(ts3);

#### 3 格式化时间

SimpleDateFormat sf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
System.out.println(sf.format(new Date())); // output:2019-08-16 21:46:22

SimpleDateFormat 构造参数的含义，请参考以下表格信息：

|  |  |  |
| --- | --- | --- |
| **字符** | **含义** | **示例** |
| y | 年 | yyyy-1996 |
| M | 月 | MM-07 |
| d | 月中的天数 | dd-02 |
| D | 年中的天数 | 121 |
| E | 星期几 | 星期四 |
| H | 小时数（0-23） | HH-23 |
| h | 小时数（1-12） | hh-11 |
| m | 分钟数 | mm-02 |
| s | 秒数 | ss-03 |
| Z | 时区 | +0800 |

使用示例：

* 获取星期几：new SimpleDateFormat(“E”).format(new Date())
* 获取当前时区：new SimpleDateFormat(“Z”).format(new Date\*())

**注意事项** ：在多线程下 SimpleDateFormat 是非线程安全的，因此在使用 SimpleDateFormat 时要注意这个问题。在多线程下，如果使用不当，可能会造成结果不对或内存泄漏等问题。

#### 4 时间转换

SimpleDateFormat sf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
// String 转 Date  
String str = "2019-10-10 10:10:10";  
System.out.println(sf.parse(str));  
//时间戳的字符串 转 Date  
String tsString = "1556788591462";  
// import java.sql  
Timestamp ts = new Timestamp(Long.parseLong(tsString)); // 时间戳的字符串转 Date  
System.out.println(sf.format(ts));

**注意事项** ：当使用 SimpleDateFormat.parse() 方法进行时间转换的时候，SimpleDateFormat 的构造函数必须和待转换字符串格式一致。

#### 5 获得昨天此刻时间

Calendar calendar = Calendar.getInstance();  
calendar.add(Calendar.DATE, -1);  
System.out.println(calendar.getTime());

### JDK 8 时间操作

JDK 8 对时间操作新增了三个类：LocalDateTime、LocalDate、LocalTime。

* LocalDate 只包含日期，不包含时间，不可变类，且线程安全。
* LocalTime 只包含时间，不包含日期，不可变类，且线程安全。
* LocalDateTime 既包含了时间又包含了日期，不可变类，且线程安全。

**线程安全性**

值得一提的是 JDK 8 中新增的这三个时间相关的类，都是线程安全的，这极大地降低了多线程下代码开发的风险。

#### 1 获取时间

// 获取日期  
LocalDate localDate = LocalDate.now();  
System.out.println(localDate); // output:2019-08-16  
// 获取时间  
LocalTime localTime = LocalTime.now();  
System.out.println(localTime); // output:21:09:13.708  
// 获取日期和时间  
LocalDateTime localDateTime = LocalDateTime.now();  
System.out.println(localDateTime); // output:2019-08-16T21:09:13.708

#### 2 获取时间戳

long milli = Instant.now().toEpochMilli(); // 获取当前时间戳（精确到毫秒）  
long second = Instant.now().getEpochSecond(); // 获取当前时间戳（精确到秒）  
System.out.println(milli); // output:1565932435792  
System.out.println(second); // output:1565932435

#### 3 时间格式化

// 时间格式化①  
DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");  
String timeFormat = dateTimeFormatter.format(LocalDateTime.now());  
System.out.println(timeFormat); // output:2019-08-16 21:15:43  
// 时间格式化②  
String timeFormat2 = LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));  
System.out.println(timeFormat2); // output:2019-08-16 21:17:48

#### 4 时间转换

String timeStr = "2019-10-10 06:06:06";  
LocalDateTime dateTime = LocalDateTime.parse(timeStr,DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));  
System.out.println(dateTime);

#### 5 获得昨天此刻时间

LocalDateTime today = LocalDateTime.now();  
LocalDateTime yesterday = today.plusDays(-1);  
System.out.println(yesterday);

### 相关面试题

#### 1. 获取当前时间有几种方式？

答：获取当前时间常见的方式有以下三种：

* new Date()
* Calendar.getInstance().getTime()
* LocalDateTime.now()

#### 2. 如何获取昨天此刻的时间？

答：以下为获取昨天此刻时间的两种方式：

// 获取昨天此刻的时间（JDK 8 以前）  
Calendar c = Calendar.getInstance();  
c.add(Calendar.DATE,-1);  
System.out.println(c.getTime());  
// 获取昨天此刻的时间（JDK 8）  
LocalDateTime todayTime = LocalDateTime.now();  
System.out.println(todayTime.plusDays(-1));

#### 3. 如何获取本月的最后一天？

答：以下为获取本月最后一天的两种方式：

// 获取本月的最后一天（JDK 8 以前）  
Calendar ca = Calendar.getInstance();  
ca.set(Calendar.DAY\_OF\_MONTH, ca.getActualMaximum(Calendar.DAY\_OF\_MONTH));  
System.out.println(ca.getTime());  
// 获取本月的最后一天（JDK 8）  
LocalDate today = LocalDate.now();  
System.out.println(today.with(TemporalAdjusters.lastDayOfMonth()));

#### 4. 获取当前时间的时间戳有几种方式？

答：以下为获取当前时间戳的几种方式：

* System.currentTimeMillis()
* new Date().getTime()
* Calendar.getInstance().getTime().getTime()
* Instant.now().toEpochMilli()
* LocalDateTime.now().toInstant(ZoneOffset.of(“+8”)).toEpochMilli()

其中，第四种和第五种方式是 JDK 8 才新加的。

#### 5. 如何优雅地计算两个时间的相隔时间？

答：JDK 8 中可以使用 Duration 类来优雅地计算两个时间的相隔时间，代码如下：

LocalDateTime dt1 = LocalDateTime.now();  
LocalDateTime dt2 = dt1.plusSeconds(60);  
Duration duration = Duration.between(dt1, dt2);  
System.out.println(duration.getSeconds()); // output:60

#### 6. 如何优雅地计算两个日期的相隔日期？

答：JDK 8 中可以使用 Period 类来优雅地计算两个日期的相隔日期，代码如下：

LocalDate d1 = LocalDate.now();  
LocalDate d2 = d1.plusDays(2);  
Period period = Period.between(d1, d2);  
System.out.println(period.getDays()); //output:2

#### 7. SimpleDateFormat 是线程安全的吗？为什么？

答：SimpleDateFormat 是非线程安全的。因为查看 SimpleDateFormat 的源码可以得知，所有的格式化和解析，都需要通过一个中间对象进行转换，这个中间对象就是 Calendar，这样的话就造成非线程安全。试想一下当我们有多个线程操作同一个 Calendar 的时候后来的线程会覆盖先来线程的数据，那最后其实返回的是后来线程的数据，因此 SimpleDateFormat 就成为了非线程的了。

#### 8. 怎么保证 SimpleDateFormat 的线程安全？

答：保证 SimpleDateFormat 线程安全的方式如下：

* 使用 Synchronized，在需要时间格式化的操作使用 Synchronized 关键字进行包装，保证线程堵塞格式化；
* 手动加锁，把需要格式化时间的代码，写到加锁部分，相对 Synchronized 来说，编码效率更低，性能略好，代码风险较大（风险在于不要忘记在操作的最后，手动释放锁）；
* 使用 JDK 8 的 DateTimeFormatter 替代 SimpleDateFormat。

#### 9. JDK 8 中新增的时间类都有哪些优点？

答：JDK 8 中的优点具体有以下几个优点，如下：

* 线程安全性
* 使用的便利性（如获取当前时间戳的便利性、增减日期的便利性等）
* 编写代码更简单优雅，如当前时间的格式化：LocalDateTime.now().format(DateTimeFormatter.ofPattern(“yyyy-MM-dd HH:mm:ss”));

#### 10. 如何比较两个时间（Date）的大小？

答：时间比较有以下三种方式：

* 获取两个时间的时间戳，得到两个 long 类型的变量，两个变量相减，通过结果的正负值来判断大小；
* 通过 Date 自带的 before()、after()、equals() 等方法比较，代码示例 date1.before(date2)；
* 通过 compareTo() 方法比较，代码示例：date1.compareTo(date2)，返回值 -1 表示前一个时间比后一个时间小，0 表示两个时间相等，1 表示前一个时间大于后一个时间。

### 总结

JDK 8 之前使用 java.util.Date 和 java.util.Calendar 来操作时间，它们有两个很明显的缺点，第一，非线程安全；第二，API 调用不方便。JDK 8 新增了几个时间操作类 java.time 包下的 LocalDateTime、LocalDate、LocalTime、Duration（计算相隔时间）、Period（计算相隔日期）和 DateTimeFormatter，提供了多线程下的线程安全和易用性，让我们可以更好的操作时间。

[点击此处下载本讲源码](https://github.com/vipstone/java-%20interview/tree/master/interview-code/src/main/java/com/interview)

更多java学习资料如下：

[Java高级架构师课程(总共122门课程，1460GB)](https://www.consultdog.com/index.php/front/44/)

[最强java面试视频课程(21门课程，126GB)](https://www.consultdog.com/index.php/front/78/)