Date/Time API

Java 8通过发布新的Date-Time API (JSR 310)来进一步加强对日期与时间的处理。对日期与时间的操作一直是Java程序员最痛苦的地方之一。标准的 java.util.Date以及后来的java.util.Calendar一点没有改善这种情况（可以这么说，它们一定程度上更加复杂）。

这种情况直接导致了Joda-Time——一个可替换标准日期/时间处理且功能非常强大的Java API的诞生。Java 8新的Date-Time API (JSR 310)在很大程度上受到Joda-Time的影响，并且吸取了其精髓。

LocalDate类

LocaleDate只持有ISO-8601格式且无时区信息的日期部分

```
LocalDate date = LocalDate.now();  // 当前日期
date = date.plusDays(1);           // 增加一天
date = date.plusMonths(1);         // 增加一月
date = date.minusDays(1);          // 减少一天
date = date.minusMonths(1);        // 减少一月

System.out.println(date);          // 依然输出当前时间  2018-12-18
```

LocalTime类

LocaleTime只持有ISO-8601格式且无时区信息的时间部分

```
LocalTime time = LocalTime.now();    // 当前时间
time = time.plusMinutes(1);          // 增加一分钟
time = time.plusSeconds(1);          // 增加一秒
time = time.minusMinutes(1);         // 减少一分钟
time = time.minusSeconds(1);         // 减少一秒

System.out.println(time);            // 依然输出当前时间  21:07:25.304
```

LocaleDateTime类和格式化

LocaleDateTime把LocaleDate与LocaleTime的功能合并起来，它持有的是ISO-8601格式无时区信息的日期与时间。

```
LocalDateTime localDateTime = LocalDateTime.now();
System.out.println(localDateTime);      // 2018-12-18T21:13:07.465, UTC格式
System.out.println(localDateTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));  // 2018-12-18 21:13:07 自定义格式
```

ZonedDateTime类

如果你需要特定时区的日期/时间，那么ZonedDateTime是你的选择。它持有ISO-8601格式具具有时区信息的日期与时间。

```
final ZonedDateTime zonedDatetimeFromZone = ZonedDateTime.now( ZoneId.of( "America/Los_Angeles" ) );
System.out.println(zonedDatetimeFromZone);        //  2018-12-18T05:15:34.486-08:00[America/Los_Angeles]
```

Clock类

它通过指定一个时区，然后就可以获取到当前的时刻，日期与时间。Clock可以替换System.currentTimeMillis()与TimeZone.getDefault()。

```
        final Clock utc = Clock.systemUTC();  // 协调世界时，又称世界统一时间、世界标准时间、国际协调时间
        final Clock shanghai = Clock.system(ZoneId.of("Asia/Shanghai"));  // 上海

        System.out.println(LocalDateTime.now(utc));          // 2018-12-18T13:18:09.176
        System.out.println(LocalDateTime.now(shanghai));     // 2018-12-18T21:18:09.177
```

Duration类

Duration使计算两个日期间的不同变的十分简单。

```
final LocalDateTime from = LocalDateTime.parse("2018-12-17 18:50:50", DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));

final LocalDateTime to = LocalDateTime.parse("2018-12-18 19:50:50", DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));

final Duration duration = Duration.between(from, to);

System.out.println("Duration in days: " + duration.toDays()); // 1

System.out.println("Duration in hours: " + duration.toHours()); // 25
```