先取り Java SE 8

Author : Acroquest Technology 岡田 拓也

第3回 これまでがいかに面倒だったか…「日時API」の改良が喜ばしい

第1回と第2回ではJava SE 8（以下Java8）の新機能であるラムダ式とストリームAPI（Application Programming Interface）を紹介しました。今回は、Java8で追加された新しい日時API（Date and Time API）を紹介します。ラムダ式とストリームAPIでは、コーディングスタイルが一変しています。一方、新しい日時APIを使う際に大きなコーディングスタイルの変更は必要ありません。

そしてJava7以前の日時APIであるDate/Calendarを捨てて、完全に1から再設計・再実装したAPIが用意されており、既存の様々な問題点が解消されています。かゆいところに手が届く改良が施されているので、ぜひ知って使いこなしましょう。

欠点の多い既存のDate/Calendarクラス

新しい日時APIが導入された背景には、既存の日付クラス（java.util.Dateとjava.util.Calendar）に欠点が多かったことが挙げられます。例えば、(1)年、月、日の数字を指定してDate/Calendarクラスのインスタンスを生成できない、または非推奨（deprecated）になっている、(2) Dateクラスでは、年、月、日の各フィールドの値を個別に取得する処理が非推奨、(3) Dateクラスでは年月日の計算ができない、(4) DateクラスとCalendarクラスは共に作成後に状態を変更できてしまう（イミュータブルでない）、といった点などです。

これらの特徴や制約のために、日付の計算にはCalendarクラスを用い、日付の状態を保持するにはDateクラスを使う、というような使い分けが必要であり、相互に変換しなければならないことが、Javaで日付を扱う際の処理を複雑にしていたといえるでしょう。また、DateとCalendarがイミュータブルではないので、意図せず値が変更されないように配慮しなければなりませんでした。

既存の日付APIの問題を回避するため、現場では時刻を扱う場合は「1970年1月1日午前0時（GMT）から経過し

リスト1 ●新しい日時API「LocalDate」「LocalTime」「LocalDateTime」（List1.java）

```
import java.time.LocalDate;
import java.time.LocalTime;                    (1)
import java.time.LocalDateTime;

public class List1 {
  public static void main(String... args) {
    // 日付
    LocalDate date = LocalDate.now();          (2)
    System.out.println(date);

    // 時間
    LocalTime time = LocalTime.now();          (3)
    System.out.println(time);

    // 日時
    LocalDateTime dateTime = LocalDateTime.now();  (4)
    System.out.println(dateTime);
  }
}
```

図1 ●リスト1の実行結果

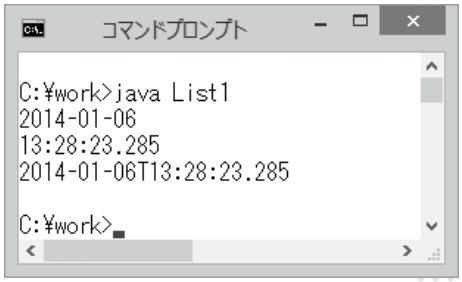

表1 ●LocalDate、LocalTime、LocalDateTimeが保持する情報

クラス	保持する情報
LocalDateクラス	日付（年、月、日）
LocalTimeクラス	時間（時、分、秒、ナノ秒）
LocalDateTimeクラス	日付（年、月、日）と時間（時、分、秒、ナノ秒）