




使用 **LocalDate**

- 使用 **LocalDate** 可以取得以下問題的答案
 - 某個日期屬於過去或未來？
 - 是否是閏年 (leap year)？
 - 是一週裡面的哪一天？
 - 是一個月裡的哪一天？
 - 下周二是哪一天？
- 過去 **java.util.Date** 類別包含時間，而程式開發者有時會使用 午夜12點 (midnight) 來表現某一天。而某些時區在日光節約時間的那一天是沒有午夜12點的。



Java 8 之前的 Date & Time

java.util.Date (Calendar, DateFormat) 的缺點：

- 不支援 流暢 (fluent) API 的使用
- Instance 都是 mutable
- 非 thread-safe
- API 種類不多



Java 8 的 Date & Time

- 相關類別和方法的使用相當直覺化
- 支援 流暢 (fluent) API 的使用方式
- Instance 都是 **immutable**
- 以 ISO 標準定義日期和時間
- Thread-safe
- 方便開發者自行擴充

和過去的日期/時間API做比較

Java 8 :

```
private static void newWay() {  
    LocalDate nowDate = LocalDate.now();  
    System.out.println(nowDate);  
  
    LocalDateTime nowDateTime = LocalDateTime.now();  
    System.out.println(nowDateTime);  
  
    LocalDate jan = LocalDate.of(2015, Month.JANUARY, 1)  
        .plusDays(5)  
        .minusDays(1);    //fluent way  
    System.out.println(jan);  
  
    DateTimeFormatter sf = DateTimeFormatter.ofPattern("hh:mm");  
    //System.out.println(sf.format(nowDate));  
    System.out.println(sf.format(nowDateTime));  
}
```

和過去的日期/時間API做比較

Java 8 前：

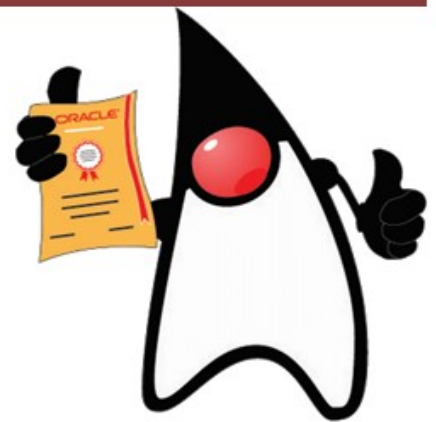
```
private static void oldWay() {  
    Date nowDate = new Date();  
    System.out.println(nowDate);  
  
    Date nowDateTime = new Date();  
    System.out.println(nowDateTime);  
  
    Calendar c = Calendar.getInstance();  
    c.set(2015, Calendar.JANUARY, 1);  
    c.add(Calendar.DATE, 5);  
    c.add(Calendar.DATE, -1);  
    Date jan = c.getTime();  
    System.out.println(jan);  
  
    SimpleDateFormat sf = new SimpleDateFormat("hh:mm");  
    System.out.println(sf.format(nowDate));  
    System.out.println(sf.format(nowDateTime));  
}
```



END ~~

Thank you!!





3/4

`java.time.format.DateTimeFormatter`

使用 DateTimeFormatter 格式化日期/時間 -1

- 使用format()方法將Date/Time 轉換為字串

```
LocalDateTime now = LocalDateTime.now();
DateTimeFormatter formatter = null;

formatter = DateTimeFormatter.ISO_LOCAL_DATE;
System.out.println(now.format(formatter));
// 2016-03-23

formatter = DateTimeFormatter.ISO_ORDINAL_DATE;
System.out.println(now.format(formatter));
// 2016-083 (days of year)

formatter = DateTimeFormatter.ISO_LOCAL_DATE_TIME;
System.out.println(now.format(formatter));
// 2016-03-23T11:20:21.065
```


使用 DateTimeFormatter 格式化日期/時間 -2

- FormatStyle 是 enum 具有 SHORT, MEDIUM, LONG, FULL.

```
formatter = DateTimeFormatter.ofPattern("EEEE, MMMM dd, yyyy G, hh:mm a");
```

```
// 4碼 表示不縮寫
```

```
System.out.println(now.format(formatter));
```

```
// 星期三, 三月 23, 2016 西元, 11:20 上午
```

```
formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.n");
```

```
System.out.println(now.format(formatter));
```

```
// 2016-03-23 11:20:21.65000000
```

```
formatter = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.MEDIUM);
```

```
System.out.println(now.format(formatter));
```

```
// 2016/3/23 上午 11:20:21
```

解析 Dates and Times

- 使用parse()方法將字串轉換為Date/Time

```
public static void main(String[] args) {  
    DateTimeFormatter dFormat = DateTimeFormatter.ofPattern("MM dd yyyy");  
    LocalDate date = LocalDate.parse("01 02 2015", dFormat);  
    System.out.println(date);  
  
    DateTimeFormatter tFormat = DateTimeFormatter.ofPattern("HH mm ss");  
    LocalTime time = LocalTime.parse("11 22 57", tFormat);  
    System.out.println(time);  
}
```



4/4

Date & Time 相關類別在Java 8前後的比較