Java8 Time API - Instant 사용법

자바8 Time API의 `Instant` 클래스는 시간을 타임스탬프로 다루기 위해서 사용합니다. 타임스탬프는 UTC 기준으로 1970년 1월 1일 0시 0분 0초를 숫자 `0`으로 정하고 그로 부터 경과된 시간을 양수 또는 음수로 표현합니다. 타임스탬프는 인간에게는 직관적이지 않은 시간의 표현 방법이지만 고전적으로 기계에게는 매우 친화적인 방법으로 현재까지 널리 사용되어 왔습니다. 일단 시간을 표현하기 위해서 별도의 타입없이 기본 데이터 타입으로 표현이 가능한데다가, 타임존이 UTC로 고정되어 있기 때문에 타임스탬프가 어느 타임존 기준인지를 고려할 필요가 없기 때문입니다.

Timestamp 나타내기

```
import java.sql.Timestamp;
import java.time.Instant;
import java.util.Calendar;
import java.util.Date;

public class Java8_Instant1 {
    // Timestamp 나타내기
    public static void main(String[] args) {
        // Java7 이전
        Timestamp timestamp1 = new Timestamp(System.currentTimeMillis());
        System.out.println(timestamp1);

        Date date = new Date();
        Timestamp timestamp2 = new Timestamp(date.getTime());
        System.out.println(timestamp2);

        Timestamp timestamp3 = new
Timestamp(Calendar.getInstance().getTimeInMillis());
        System.out.println(timestamp3);
        System.out.println();

        // Java8 이후
        Instant instant1 = Instant.now();
        Instant instant2 = timestamp1.toInstant();
        Instant instant3 = date.toInstant();
        System.out.println(instant1);
        System.out.println(instant2);
        System.out.println(instant3);
        System.out.println();

        Timestamp timestamp4 = Timestamp.from(instant1);
        System.out.println(timestamp4);
    }
}
```

결과

```
2021-10-08 11:20:37.202
2021-10-08 11:20:37.217
2021-10-08 11:20:37.262

2021-10-08T02:20:37.325662200Z
2021-10-08T02:20:37.202Z
2021-10-08T02:20:37.217Z

2021-10-08 11:20:37.3256622
```

Instant 객체만들기

UTC 기준시인 1970년 1월 1일 0시 0분 0초에 해당하는 Instant 객체는 Instant.EPOCH라는 정적 필드에 저장되어 있습니다.
그리고 Instant.ofEpochSecond() 정적 메소드를 통해서 타임스탬프를 나타내는 Instant 객체를 만들 수 있습니다.

```
import java.time.Instant;
public class Java8_Instant2 {
    public static void main(String[] args) {
        Instant instant1 = Instant.EPOCH;
        Instant instant2 = Instant.ofEpochSecond(0);
        Instant instant3 = Instant.MIN;
        Instant instant4 = Instant.MAX;
        Instant instant5 = Instant.now();
        Instant instant6 = Instant.ofEpochMilli(System.currentTimeMillis());
        Instant instant7 =
Instant.ofEpochSecond(System.currentTimeMillis()/1000);
        Instant instant8 = Instant.ofEpochSecond(16_0000_0000L);
        Instant instant9 = Instant.ofEpochSecond(-16_0000_0000L);

        System.out.println(instant1);
        System.out.println(instant2);
        System.out.println(instant3);
        System.out.println(instant4);
        System.out.println(instant5);
        System.out.println(instant6);
        System.out.println(instant7);
        System.out.println(instant8);
        System.out.println(instant9);
    }
}
```

결과

```
1970-01-01T00:00:00Z
1970-01-01T00:00:00Z
-1000000000-01-01T00:00:00Z
+1000000000-12-31T23:59:59.999999999Z
2021-10-08T02:23:31.083176400Z
2021-10-08T02:23:31.083Z
2021-10-08T02:23:31Z
2020-09-13T12:26:40Z
1919-04-20T11:33:20Z
```

위의 instant8과 instant9를 보면 16억과 -16억이 각각 2020-09-13T12:26:40Z 와 1919-04-20T11:33:20Z를 나타냄을 알 수 있습니다. 타임스탬프 값은 사람이 눈으로만 봤을 때는 계산기 없이는 바로 시간을 파악하기가 어렵기 때문에 Instant 객체로 변환해서 다루면 편리합니다.

현재 시간의 타임스탬프 값을 구하기

`Instant.now()` 정적 메소드를 호출하면 현재 시간의 `Instant` 객체를 얻을 수 있습니다. `Instant` 객체는 UTC 기준의 ISO 포멧으로 출력됩니다. 그리고 `Instant` 객체에 `getEpochSecond()`나 `toEpochMilli()` 메소드를 호출하여 초단위 또는 밀리 초 단위 타임스탬프 값을 `long` 타입으로 얻을 수 있습니다.

```
import java.sql.Timestamp;
import java.time.Instant;
import java.util.Date;
import com.google.gson.Gson;

public class Java8_Instant3 {
    // 현재 시간의 타임스탬프 값을 구하기
    public static void main(String[] args) {
        Instant instant = Instant.now();
        System.out.println("Current Instant : "+ instant);

        long epochSecond = instant.getEpochSecond();
        System.out.println("Current Timestamp in seconds : " + epochSecond);

        long epochMilli = instant.toEpochMilli();
        System.out.println("Current Timestamp in milli seconds : " +
epochMilli);

        System.out.println("getNano() : " + instant.getNano());
        System.out.println();

        Gson gson = new Gson();

        // {"seconds":1633661849,"nanos":15688400} : 두개로 나누어져 있다.
        System.out.println(gson.toJson(instant));
        // "Oct 8, 2021, 11:59:20 AM"
        Timestamp timestamp = Timestamp.from(instant);
        System.out.println(gson.toJson(timestamp));
        // "Oct 8, 2021, 12:01:29 PM"
        Date date = new Date(epochMilli);
        System.out.println(gson.toJson(date));
    }
}
```

결과

```
Current Instant : 2021-10-08T03:19:54.422138200Z
Current Timestamp in seconds : 1633663194
Current Timestamp in milli seconds : 1633663194422
getNano() : 422138200

{"seconds":1633663194,"nanos":422138200}
"Oct 8, 2021, 12:19:54 PM"
"Oct 8, 2021, 12:19:54 PM"
```

Instant와 ZonedDateTime 간 상호 변환하기

`Instant` 클래스의 `atZone()` 메서드와 `ZonedDateTime` 클래스의 `toInstnat()` 메소드를 통해서 두 타입의 객체는 서로 변환이 가능합니다. 예를 들어 2018년 동계 올림픽 개막식은 한국 시간으로 2월 9일 20시 00분에 치뤄졌으며 이를 서울 타임존의 `ZonedDateTime` 객체로 나타낼 수 있습니다. 이를 `Instant` 객체로 변환하면 UTC 기준 시간이 되기 때문에 9시간 느린 11:00분이 되고, 타임스탬프 값은 `1518174000`이 됩니다. 다시 이 `Instant` 객체를 벤쿠버 타임존의 `ZonedDateTiem` 객체로 변환하면 7시간 더 느린 03:00분이 되게 됩니다.

```
import java.time.Instant;
import java.time.Year;
import java.time.ZoneId;
import java.time.ZonedDateTime;

public class Java8_Instant4 {
    // Instant와 ZonedDateTime 간 상호 변환하기
    public static void main(String[] args) {
        ZonedDateTime zdtSeoul = Year.of(2018).atMonth(2).atDay(9).atTime(20,
00).atZone(ZoneId.of("Asia/Seoul"));
        System.out.println("Time in Seoul = " + zdtSeoul);

        Instant instant = zdtSeoul.toInstant();
        System.out.println("Instant = " + instant + ", Timestamp = " +
instant.getEpochSecond());

        ZonedDateTime zdtVancouver =
instant.atZone(ZoneId.of("America/Vancouver"));
        // ZonedDateTime zdtVancouver = ZonedDateTime.ofInstant(instant,
ZoneId.of("America/Vancouver")); 와 동일
        System.out.println("Tine in Vancouver = " + zdtVancouver);
    }
}
```

결과

```
Time in Seoul = 2018-02-09T20:00+09:00[Asia/Seoul]
Instant = 2018-02-09T11:00:00Z, Timestamp = 1518174000
Tine in Vancouver = 2018-02-09T03:00-08:00[America/Vancouver]
```

그 밖에 유용한 메소드들

`Instant`는 타임스탬프를 표현하는 클래스 답게 초나 밀리 초 단위로 시간을 더하거나 빼는 메소드들을 제공합니다. 또한 시간 비교를 위한 메소드도 제공합니다.

```
import java.time.Duration;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class Java8_Instant5 {
    // 그 밖에 유용한 메소드들
    public static void main(String[] args) {
        Instant instant = Instant.now();
        System.out.println("현재 : " + instant);
        System.out.println("100초 후 : " + instant.plusSeconds(100));
        System.out.println("100초 전 : " + instant.minusSeconds(100));

        System.out.println("4분 후 : " + instant.plus(Duration.ofMinutes(4)));
        System.out.println("4분 전 : " + instant.minus(Duration.ofMinutes(4)));

        System.out.println("10일 후 : " + instant.plus(10, ChronoUnit.DAYS));
        System.out.println("10일 전 : " + instant.minus(10,ChronoUnit.DAYS));
        System.out.println();

        Instant instant2 = Instant.parse("2021-10-18T03:45:09.652967800Z");
        System.out.println("instant : " + instant);
        System.out.println("instant2 : " + instant2);
        System.out.println(instant.isBefore(instant2));
        System.out.println(instant.isAfter(instant2));

    }
}
```

결과

```
현재 : 2021-10-08T03:48:50.962471600Z
100초 후 : 2021-10-08T03:50:30.962471600Z
100초 전 : 2021-10-08T03:47:10.962471600Z
4분 후 : 2021-10-08T03:52:50.962471600Z
4분 전 : 2021-10-08T03:44:50.962471600Z
10일 후 : 2021-10-18T03:48:50.962471600Z
10일 전 : 2021-09-28T03:48:50.962471600Z

instant : 2021-10-08T03:48:50.962471600Z
instant2 : 2021-10-18T03:45:09.652967800Z
true
false
```