Cập nhật tháng 8 năm 2024

[Bài Đọc] Datetime API

DateTime API được giới thiệu trong **Java 8** như một phần của gói **java.time.** Nó cải thiện đáng kể việc xử lý ngày, giờ, múi giờ và các phép tính liên quan, khắc phục nhiều hạn chế của các lớp cũ như Date, Calendar

1. **Tại sao cần DateTime API mới?**

- Trước Java 8, việc xử lý ngày và giờ thường sử dụng các lớp như Date, Calendar và SimpleDateFormat. Những lớp này có nhiều hạn chế:
  - Không bất biến (mutable), dễ dẫn đến lỗi khi làm việc với nhiều luồng
  - API khó sử dụng và thiếu rõ ràng
  - Thiếu hỗ trợ tốt cho múi giờ và các phép tính phức tạp
- DateTime API trong Java 8 giải quyết những vấn đề này với một cách tiếp cận hiện đại, nhất quán và bất biến

2. **Các lớp chính trong DateTime API**

- **LocalDate**
  - Đại diện cho một ngày mà không có thông tin về thời gian hoặc múi giờ
  - Ví dụ: 2025-01-01
  - Cách sử dụng:

```
public static void main(String[] args)  new *
{
    LocalDate today = LocalDate.now();
    System.out.println("Hôm nay là: " + today);

    LocalDate specificDate = LocalDate.of( year: 2023,  month: 12,  dayOfMonth: 25);
    System.out.println("Ngày cụ thể: " + specificDate);
}
```

- **LocalTime**
  - Đại diện cho thời gian trong ngày mà không có thông tin ngày hoặc múi giờ
  - Ví dụ: 10:15:30
  - Cách sử dụng:

```java
public static void main(String[] args) {
    LocalTime currentTime = LocalTime.now();
    System.out.println("Thời gian hiện tại: " + currentTime);

    LocalTime specificTime = LocalTime.of( hour: 14, minute: 30, second: 0);
    System.out.println("Thời gian cụ thể: " + specificTime);
}
```

- **LocalDateTime**
  - Kết hợp cả ngày và giờ mà không có múi giờ
  - Ví dụ: 2025-01-01T10:15:30
  - Cách sử dụng:

```java
public static void main(String[] args) {
    LocalDateTime now = LocalDateTime.now();
    System.out.println("Ngày giờ hiện tại: " + now);

    LocalDateTime specificDateTime = LocalDateTime.of( year: 2025, month: 1, dayOfMonth: 1, hour: 10, minute: 15);
    System.out.println("Ngày giờ cụ thể: " + specificDateTime);
}
```

- **ZonedDateTime**
  - Đại diện cho ngày, giờ cùng với thông tin múi giờ.
  - Cách sử dụng:

```java
public static void main(String[] args) {
    ZonedDateTime zonedDateTime = ZonedDateTime.now();
    System.out.println("Ngày giờ với múi giờ: " + zonedDateTime);

    ZonedDateTime specificZone = ZonedDateTime.now(ZoneId.of( zoneId: "Asia/Ho_Chi_Minh"));
    System.out.println("Ngày giờ theo múi giờ Việt Nam: " + specificZone);
}
```

- **Instant**
  - Đại diện cho một thời điểm (timestamp) trong dòng thời gian UTC
  - Cách sử dụng:

```java
public static void main(String[] args) {
    Instant now = Instant.now();
    System.out.println("Thời điểm hiện tại: " + now);
}
```

- **Period và Duration**
  - **Period**: Được sử dụng để đại diện khoảng cách giữa các ngày (ngày, tháng, năm)
  - **Duration**: Được sử dụng để đại diện khoảng cách giữa các thời điểm (giờ, phút, giây, nano giây)

```java
public static void main(String[] args) new *
{
    LocalDate startDate = LocalDate.of( year: 2023,  month: 1,  dayOfMonth: 1);
    LocalDate endDate = LocalDate.of( year: 2025,  month: 1,  dayOfMonth: 1);

    Period period = Period.between(startDate, endDate);
    System.out.println("Khoảng cách: " + period.getYears() + " năm " + period.getMonths() + " tháng " + period.getDays() + " ngày");
}
```

3. Định dạng và phân tích (Formatting and Parsing)

- Lớp DateTimeFormatter hỗ trợ định dạng và phân tích ngày, giờ theo định dạng cụ thể

```java
public static void main(String[] args)  new *
{
    LocalDateTime now = LocalDateTime.now();
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

    String formattedDate = now.format(formatter);
    System.out.println("Định dạng ngày giờ: " + formattedDate);

    LocalDateTime parsedDate = LocalDateTime.parse( text: "10-01-2025 12:00:00", formatter);
    System.out.println("Phân tích từ chuỗi: " + parsedDate);
}
```

4. Ưu điểm của DateTime API

- **Bất biến (Immutable)**: Đảm bảo an toàn trong các ứng dụng đa luồng
- **Dễ sử dụng**: API rõ ràng và trực quan
- **Hỗ trợ múi giờ tốt**: Dễ dàng xử lý các múi giờ khác nhau
- **Tích hợp tốt với Java 8+**: Sử dụng kết hợp với Stream API và các tính năng khác

Link tài nguyên đọc thêm: https://www.baeldung.com/java-8-date-time-intro

Danh sách các bài học