Java 日期时间与正则表达式

1、java.time.LocalDate类表示日期；

你可以使用该类的**now()**方法获取当前日期，或者使用**of()**方法创建一个指定日期的实例，例如：

Java 复制代码

```java
LocalDate today = LocalDate.now();
LocalDate myBirthday = LocalDate.of(2000, Month.JANUARY, 1);
```

2、java.time.LocalTime类表示时间；

你可以使用该类的**now()**方法获取当前时间，或者使用**of()**方法创建一个指定时间的实例，例如：

Java 复制代码

```java
LocalTime now = LocalTime.now();
LocalTime sixThirty = LocalTime.of(6, 30);
```

3、java.time.LocalDateTime类表示日期和时间；

你可以使用该类的**now()**方法获取当前日期和时间，或者使用**of()**方法创建一个指定日期和时间的实例，例如：

```java
LocalDateTime now = LocalDateTime.now();
LocalDateTime dateTime = LocalDateTime.of(2023, Month.JANUARY, 1, 6, 30);
```

4、java.time.format.DateTimeFormatter类用于格式化日期和时间；

你可以使用该类的**ofPattern()**方法创建一个格式化模板，然后使用该模板的**format()**方法格式化日期和时间，例如：

```java
LocalDateTime now = LocalDateTime.now();
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
String formattedDateTime = now.format(formatter);
```

5、创建正则表达式对象

可以使用**java.util.regex.Pattern**类创建正则表达式对象，该类提供了**compile()**方法来编译正则表达式，并返回一个**Pattern**对象。例如：

```java
Pattern pattern = Pattern.compile("regex");
```

6、匹配字符串

可以使用**Matcher**类来匹配字符串，并执行相应的操作，例如：可以使用**matches()**方法来检查给定的字符串是否与正则表达式匹配

```java
String input = "example string";
Pattern pattern = Pattern.compile("example.*");
Matcher matcher = pattern.matcher(input);
if (matcher.matches()) {
    System.out.println("Match found");
}
```

7、查找匹配

可以使用**find()**方法在给定的输入中查找正则表达式的匹配项。例如：

```java
String input = "example string";
Pattern pattern = Pattern.compile("example.*");
Matcher matcher = pattern.matcher(input);
if (matcher.find()) {
    System.out.println("Match found at index " + matcher.start());
}
```

8、替换字符串

可以使用**replaceAll()**方法来替换匹配正则表达式的字符串。例如：

```java
String input = "example string";
Pattern pattern = Pattern.compile("example");
Matcher matcher = pattern.matcher(input);
String output = matcher.replaceAll("replacement");
System.out.println(output);
```

9、匹配模式

Java中的正则表达式支持许多模式，可以使用这些模式来调整匹配行为。例如，可以使用**Pattern.CASE_INSENSITIVE**模式来进行不区分大小写的匹配：

```java
String input = "Example String";
Pattern pattern = Pattern.compile("example.*", Pattern.CASE_INSENSITIVE);
Matcher matcher = pattern.matcher(input);
if (matcher.matches()) {
    System.out.println("Match found");
}
```