Java Date and Time

❮ Previous

Next ❯

Java Dates

Java does not have a built-in Date class, but we can import the `java.time` package to work with the date and time API. The package includes many date and time classes. For example:

Class	Description
`LocalDate`	Represents a date (year, month, day (yyyy-MM-dd))
`LocalTime`	Represents a time (hour, minute, second and microsecond (HH-mm-se-zzz))
`LocalDateTime`	Represents both a date and a time (yyyy-MM-dd-HH-mm-ss.zzz)
`DateTimeFormatter`	Formatter for displaying and parsing date-time objects

If you don't know what a package is, read our Java Packages Tutorial.

Display Current Date

To display the current date, import the `java.time.LocalDate` class, and use its `now()` method:

Example

```
import java.time.LocalDate; // import the LocalDate class

public class MyClass {
  public static void main(String[] args) {
    LocalDate myObj = LocalDate.now(); // Create a date object
    System.out.println(myObj); // Display the current date
  }
}
```

The output will be:

```
2018-11-122018-11-12
```

Run example »

Display Current Time

To display the current time (hour, minute, second, and microsecond), import the `java.time.LocalTime` class, and use its `now()` method:

Example

```
import java.time.LocalTime; // import the LocalTime class

public class MyClass {
  public static void main(String[] args) {
    LocalTime myObj = LocalTime.now();
    System.out.println(myObj);
  }
}
```

The output will be:

```
13:37:39.92738013:37:36.284052
```

Run example »

Display Current Date and Time

To display the current date and time, import the `java.time.LocalDateTime` class, and use its `now()` method:

Example

```
import java.time.LocalDateTime; // import the LocalDateTime class

public class MyClass {
  public static void main(String[] args) {
    LocalDateTime myObj = LocalDateTime.now();
    System.out.println(myObj);
  }
}
```

The output will be:

```
2018-11-12T13:37:39.9275862018-11-12T13:37:36.330992
```

Run example »

Formatting Date and Time

The "T" in the example above is used to separate the date from the time. You can use the `DateTimeFormatter` class with the `ofPattern()` method in the same package to format or parse date-time objects. The following example will remove the "T" in the date-time:

Example

```
import java.time.LocalDateTime; // Import the LocalDateTime class
import java.time.format.DateTimeFormatter; // Import the
DateTimeFormatter class

public class MyClass {
  public static void main(String[] args) {
    LocalDateTime myDateObj = LocalDateTime.now();
    System.out.println("Before formatting: " + myDateObj);
    DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-
yyyy HH:mm:ss");

    String formattedDate = myDateObj.format(myFormatObj);
    System.out.println("After formatting: " + formattedDate);
  }
}
```

The output will be:

```
Before Formatting: 2018-11-12T13:37:39.927151Before Formatting: 2018-11-
12T13:37:36.331920
After Formatting: 2018-11-12 13:37:39After Formatting: 2018-11-12 13:37:36
```

Run example »

The `ofPattern()` method accepts all sorts of values, if you want to display the date and time in a different format. For example:

Value	Example	Tryit
yyyy-MM-dd	"1988-09-29"	Try it »
dd/MM/yyyy	"29/09/1988"	Try it »
dd-MMM-yyyy	"29-Sep-1988"	Try it »
E, MMM dd yyyy	"Mon, Sep 29 1988"	Try it »

❮ Previous Next ❯