# FECHAS EN JAVA. CLASE LocalDate

## Contenido

## 1. Introducción

**La clase Java LocalDate** se introduce en Java 8 en el paquete java.time. Una instancia de la clase LocalDate representa Una fecha sin la información de la zona horaria. Veamos ejemplo de la clase LocalDate y sus métodos.

## 2. Ejemplo de LocalDate para mostrar la fecha actual o cualquier fecha

```
import java.time.LocalDate;
public class Ejemplo1 {
   public static void main(String[] args) {

       /* Usamos el metodo now() de la clase LocalDate para
       * obtener la fecha actual
       */

      LocalDate currentDate = LocalDate.now();
      System.out.println("Hoy es: " + currentDate);

      /* Convertimos una fecha en formato LocalDate
       * con el metodo of() de la clase LocalDate
       */

      LocalDate unaFecha = LocalDate.of(2024, 01, 23);
      System.out.println("La fecha dada es: " + unaFecha);
   }
}
```

## 3. Ejemplo de LocalDate para comprobar si un año dado es bisiesto o no

El método isLeapYear nos devuelve true si el año es bisiesto y false en caso contrario.

```
import java.time.LocalDate;

public class Ejemplo2 {
	public static void main(String[] args) {
		LocalDate unaFecha1 = LocalDate.of(2017, 6, 23);
		System.out.println(unaFecha1.isLeapYear());

		LocalDate unaFecha2 = LocalDate.of(2012, 10, 20);
		System.out.println(unaFecha2.isLeapYear());

		LocalDate unaFecha3 = LocalDate.of(2000, 11, 14);
		System.out.println(unaFecha3.isLeapYear());
	}
}
```

## 4. Convierta la cadena a LocalDate

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Ejemplo4 {
  public static void main(String[] args) {
      // Escribimos el patrón con el que queremos convertir la cadena a fecha
      DateTimeFormatter formatter =
      DateTimeFormatter.ofPattern("yyyy/MM/dd");

      // Cadena que queremos convertir a fecha
      String stringDate = "2023/10/23";

      // Convertimos la cadena al formato fecha deseado
      LocalDate date = LocalDate.parse(stringDate, formatter);

      // Si lo mostramos asi, lo muestra en el formato por defecto
      System.out.println("Local Date: " + date);

      // Asi lo mostrara en el formato que indicamos en DateTimeFormatter
      System.out.println("Local Date in the given format: " +
      formatter.format(date));
  }
}
```

## 5. Métodos LocalDate

**`LocalDateTime atTime(int hour, int minute, int second)`**

Este método combina la fecha con la hora especificada para crear LocalDateTime.

hora: la hora del día en que se debe usar, de 0 a 23
minuto: los minutos de uso, de 0 a 59
segundo: el segundo del minuto a representar, de 0 a 59

**`LocalDateTime atStartOfDay ()`**

Este método funciona como el método atTime (), sin embargo, en este método no especificamos la hora, combina la hora de la medianoche (inicio del día) con la fecha para crear LocalDateTime al comienzo del día.

**`int compareTo (ChronoLocalDate otro)`**

Compara esta fecha con la otra fecha especificada. Este método devuelve un valor negativo si la fecha es menor que otra fecha y devuelve un valor positivo si la fecha es mayor que la otra fecha.

**`public boolean equals (Object obj)`**

Comprueba si esta fecha es igual a otra fecha. Este método es similar al método compareTo () excepto que devuelve el valor booleano verdadero y falso según la comparación.

**`boolean isAfter(ChronoLocalDate other)`**

Compruebe si esta fecha es posterior a la fecha especificada. Devuelve verdadero si esta fecha es posterior a la “otra fecha” especificada; de lo contrario, devuelve falso.

**`boolean isBefore(ChronoLocalDate other)`**

Compruebe si esta fecha es anterior a la fecha especificada. Devuelve verdadero si esta fecha es anterior a la “otra fecha” especificada; de lo contrario, devuelve falso.

**`boolean isEqual(ChronoLocalDate other`**

Compruebe si esta fecha es la misma que la “otra” fecha especificada. Si ambas fechas son iguales, devuelve verdadero; de lo contrario, devuelve falso.

**`static LocalDate now()`**

Este método devuelve la fecha actual.

**static LocalDate of(int year, Month month, int dayOfMonth)**

Crea la instancia de LocalDate a partir del día, mes y año especificados

**static LocalDate ofYearDay(int year, int dayOfYear)**

Obtiene una instancia de LocalDate del año y dayOfYear especificados.

**static LocalDate parse(CharSequence text, DateTimeFormatter formatter)**

Ya hemos visto el ejemplo de este método arriba. Este método convierte la cadena de fecha en LocalDate usando el formato especificado por DateTimeFormatter.

**int getYear()**

Devuelve el año de la fecha especificada por LocalDate.

**int getMonthValue()**

Devuelve el valor entero del mes entre 1 y 12.

**Month getMonth()**

Obtiene el campo del mes del año mediante la enumeración Month.

**int getDayOfMonth()**

Este método devuelve el día del mes. El valor está entre 1 y 31.

**int getDayOfYear()**

1 a 365, o 366 en un año bisiesto

**DayOfWeek getDayOfWeek()**

Obtiene el campo del día de la semana, que es una enumeración DayOfWeek.

**boolean isLeapYear()**

Ya hemos visto el ejemplo anterior del método isLeapYear (). Este método comprueba si el año especificado en LocalDate es un año bisiesto o no.

**int lengthOfYear()**

Devuelve la duración del año. El valor devuelto sería 366 si el año es bisiesto, 365 en caso contrario.

**LocalDate plusYears(long yearsToAdd)**

Devuelve LocalDate después de agregar el número especificado de años “yearsToAdd”.

**`LocalDate plusMonths(long monthsToAdd)`**

Devuelve LocalDate después de agregar el número especificado de meses "monthsToAdd".

**`LocalDate plusWeeks(long weeksToAdd)`**

Devuelve LocalDate después de agregar el objeto especificado número de semanas "weeksToAdd".

**`LocalDate plusDays(long daysToAdd)`**

Devuelve LocalDate después de agregar el número especificado de días "daysToAdd".

**`LocalDate minusYears(long yearsToSubtract)`**

Devuelve LocalDate después de restar el número especificado de años "yearsToSubtract".

**`LocalDate minusMonths(long monthsToSubtract)`**

Devuelve LocalDate después de restar el número especificado de meses "monthsToSubtract".

**`LocalDate minusWeeks(long weeksToSubtract)`**

Devuelve LocalDate después de restar el número especificado de semanas "weeksToSubtract".

**`LocalDate minusDays(long daysToSubtract)`**

Devuelve LocalDate después de restar el número especificado de días "daysToSubtract".

**Otra clase (es un enumerado en realidad) interesante que investigar y probar ver es:**

https://docs.oracle.com/en/java/javase/20/docs/api/java.base/java/time/DayOfWeek.html