¿Cómo podemos trabajar con fechas y horas en Java?

En JAVA existen dos modos de trabajar con Fechas:

- La forma antigua (anterior a Java 8): Se usan las clases java.util.Date y java.util.Calendar
- La forma nueva (posterior a Java 8): Se usan las clases java.time.LocalDate, LocalTime, LocalDateTime

Las nuevas clases son mejores porque simplifican muchos problemas y funciones. Lo que ocurre es que las antiguas aún están muy extendidas y se usan mucho. Nosotros vamos a ver las nuevas. Aunque al final también explicaremos cómo se puede crear una fecha con las clases antiguas.

¿Cómo podemos crear un objeto que sea una Fecha?

Depende de qué dato queramos manejar, tendremos estas tres clases:

- LocalDate → Para trabajar sólo con fecha sin hora
- LocalTime → Para trabajar sólo con hora sin fecha
- LocalDateTime → Para trabajar con fecha y hora

Las tres clases se usan del mismo modo. Viendo una, vemos todas.

Para crear un objeto que sea una fecha tenemos estas dos opciones:

1. Crear una fecha con la fecha de hoy:
```
LocalDate fecha = LocalDate.now();
```
2. Crear una fecha para un momento determinado:
```
LocalDate fecha = LocalDate.of(2021, Month.APRIL, 1); // 01/04/2021
```
También podríamos haber escrito el número del mes:
```
LocalDate fecha = LocalDate.of(2021, 4, 1); // 01/04/2021
```
Si indicamos unos valores que no son una fecha correcta, nos lanzará un error.

¿Cómo cambiamos una fecha que ya tenemos?

Si queremos cambiar algún valor de una fecha que ya tenemos creadas, podemos usar los métodos with():

```
LocalDate fecha = LocalDate.now();
fecha = fecha.withDayOfMonth(10); // cambiado a día 10
fecha = fecha.withMonth(3); // cambiamos a marzo
fecha = fecha.withYear(2021); // cambiamos al año 2021
```

¿Cómo obtenemos las partes de una fecha?

Una vez que tenemos un objeto fecha, podemos usar un conjunto de métodos get() para solicitar diferente información. Por ejemplo:

```
LocalDate fecha = LocalDate.now();
Integer dia = fecha.getDayOfMonth();
Integer mes = fecha.getMonthValue();
Integer año = fecha.getYear();
```

También podemos preguntar si la fecha es un año bisiesto, o la cantidad de días del mes:

```
Boolean bisiesto = fecha.isLeapYear();
```

```
Integer cantidadDiasMes = fecha.lengthOfMonth();
```

¿Cómo sumamos o restamos valores a una fecha?

Una vez que tenemos un objeto fecha, podemos usar un conjunto de métodos plus() y minus() para sumar y restar valores. Veamos ejemplos:

```
LocalDate fecha = LocalDate.now();
fecha = fecha.plusDays(4); // suma 4 días
fecha = fecha.minusMonths(1); // resta un mes
fecha = fecha.plusWeeks(10); // suma 10 semanas
fecha = fecha.minusYears(100); // resta 100 años
```

¿Cómo obtenemos el tiempo que hay entre dos fechas y compararlas?

Para ello usamos el método until() que nos devolverá un objeto de tipo Period. Será el periodo transcurrido entre las dos fechas. A ese periodo, podemos luego solicitarle que nos dé los días, meses o años.

IMPORTANTE: al llamar a until(), debemos hacerlo sobre la fecha más antigua, si lo hacemos al revés, el periodo será negativo.

```
LocalDate fecha = LocalDate.now();
LocalDate fechaAnterior = LocalDate.of(2021, Month.APRIL, 1);

Period periodo = fechaAnterior.until(fecha);

Integer añosDiferencia = periodo.getYears();
Integer mesesDiferencia = periodo.getMonths();
Integer diasDiferencia = periodo.getDays();
```

Hay que tener en cuenta que los meses de diferencia, por ejemplo, no es el total de meses. Si comparo el 01/01/2000 con 02/02/2001, el periodo será 1 año + 1 mes + 1 día. Es decir, los meses que me devuelve el periodo son 1. Si quiero saber el total de meses de diferencia, tendrá que sumarle a esos meses la cantidad de años multiplicada por 12.

Con los días totales habría que hacer algo similar.

Para comparar dos fecha, además de los métodos equals() y compare() de Object, también tenemos métodos específicos:

```
Boolean anterior = fecha.isAfter(fechaAnterior);
Boolean posterior = fecha.isBefore(fechaAnterior);
Boolean iguales = fecha.isEqual(fechaAnterior);
```

¿Cómo imprimimos u obtenemos una String de la fecha?

Si llamamos al toString() de la clase, obtendremos una representación de la fecha en formato inglés (yyyy-mm-dd)

Pero podemos cambiar el formato creando un objeto de la clase DateTimeFormater. Veamos ejemplo:

```
LocalDate fecha = LocalDate.now();
DateTimeFormatter formato = DateTimeFormatter.ofPattern("dd/MM/yyyy");
String fechaComoCadena = fecha.format(formato);
```

A la hora de indicar el patrón, tenemos que saber lo que significa cada letra. Lo más habitual es:

- dd → día del mes con dos cifras
- MM → mes del año con dos cifras
- yyyy → año con cuatro cifras
- hh → hora en formato 12 horas (las 23 se mostrarían como 11)
- HH → hora en formato 24 horas con dos dígitos
- mm → minutos con dos dígitos
- ss → segundos con dos dígitos

Podemos ver la documentación completa aquí:

https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html

¿Cómo obtenemos una fecha a partir de un String (parsear)?

De la misma manera que el caso anterior, tendremos que crear un DateTimeFormater con el patrón en el que esté escrita la fecha. Luego llamamos al método parse().

Veamos ejemplo:

```
String fechaComoCadena = "01/04/2021";
DateTimeFormatter formato = DateTimeFormatter.ofPattern("dd/MM/yyyy");
LocalDate fecha = LocalDate.parse(fechaComoCadena, formato);
```

¿Cómo se trabaja con la antigua clase java.util.Date?

Las clases antiguas son varias:

- java.util.Date: para trabajar con fecha/hora. No funcionan como un calendario como LocalDate, sino que guardan internamente el número de milisegundos transcurridos desde el 1 de enero de 1970 hasta la fecha que registran. A partir de ahí, hacen todos los cálculos.
- java.sql.Date: hereda de la anterior. Se usa para trabajar con JDBC (driver de BBDD)
- java.util.Calendar: se utiliza cuando queremos hacer operaciones de calendario (sumar días, obtener el valor del mes, etc.)

Son más complejas de usar. Veamos cómo crear una fecha:

```
// Obtener la fecha hora actual
Date fechaHoy = new Date();

// Obtener una fecha a partir de una cadena con un formato determinado
SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
Date fechaAbril = formato.parse("01/04/2021");

// Imprimir una fecha con un formato determinado
String fechaCadena = formato.format(fechaHoy);
```

¿Cómo convertir de java.util.Date a LocalDate y viceversa?

La forma más rápida es utilizar la clase java.sql.Date.

Veamos un ejemplo:

```
LocalDate localDate = LocalDate.now();

// Convertir LocalDate en java.sql.Date y luego esta en java.util.Date
java.sql.Date sqlDate = java.sql.Date.valueOf(localDate);
Date utilDate = new Date(sqlDate.getTime());

// Convertir java.util.Date en java.sql.Date y luego esta en LocalDate
sqlDate = new java.sql.Date(utilDate.getTime());
localDate = sqlDate.toLocalDate();
```