

Java String to Date es una de las conversiones más habituales que tenemos que hacer . Para ello hay que recordar que en Java 8 llegan nuevas clases inmutables orientadas a simplificar el manejo de fechas. En concreto `LocalDate` , `LocalTime` y `LocalDateTime` son las principales y simplifican sobre manera el manejo de fechas comparadas con `Date` , `DateFormat` y `Calendar` que son clásicas de Java.



Vamos a ver unos ejemplos básicos.

Java String to Date

A partir de ahora es muy sencillo convertir una cadena a fecha ya que la propia clase `LocalDate` nos admite el uso de cadenas de forma directa a través del método `parse` siguiendo un formato muy concreto de año-mes-día.

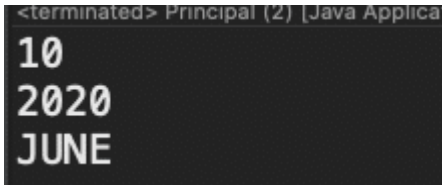
```
package com.arquitecturajava;

import java.time.LocalDate;

public class Principal {
    public static void main(String[] args) {
        LocalDate fecha = LocalDate.parse("2020-06-10");
        System.out.println(fecha.getDayOfMonth());
        System.out.println(fecha.getYear());
        System.out.println(fecha.getMonth());
    }
}
```

El método `parse` recibe el año el mes y el día separados por un guion y automáticamente

convierte la fecha, podemos ver el resultado en la consola:



```
<terminated> Principal (2) [Java Applica  
10  
2020  
JUNE
```

Java LocalTime

De igual forma podemos trabajar con LocalTime que almacena hora minutos y segundos y nos permitirá una transformación de estos datos.

```
package com.arquitecturajava;
```

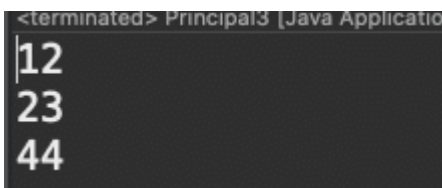
```
import java.time.LocalTime;
```

```
public class Principal3 {
```

```
    public static void main(String[] args) {  
        LocalTime localTime = LocalTime.parse("12:23:44");  
        System.out.println(localTime.getHour());  
        System.out.println(localTime.getMinute());  
        System.out.println(localTime.getSecond());  
    }
```

```
}
```

El resultado es muy similar al anterior pero orientado al uso horario:



```
<terminated> Principal3 [Java Applicatio  
12  
23  
44
```

Java 8 LocalDateTime

Java 8 también permite combinar ambos conceptos en una sola clase que aglutine el concepto de Date y de Time que como no podría ser de otra manera se denomina **LocalDateTime**. Así podremos realizar un cambio de Java String to Date aplicando tanto fecha como hora.

```
import java.time.LocalDateTime;

public class Principal2 {

    public static void main(String[] args) {
        LocalDateTime localDateTime =
LocalDateTime.parse("2020-06-01T11:20:15");
        System.out.println(localDateTime.getHour());
        System.out.println(localDateTime.getMinute());
        System.out.println(localDateTime.getSecond());
        System.out.println(localDateTime.getDayOfMonth());
        System.out.println(localDateTime.getMonthValue());
        System.out.println(localDateTime.getYear());

    }

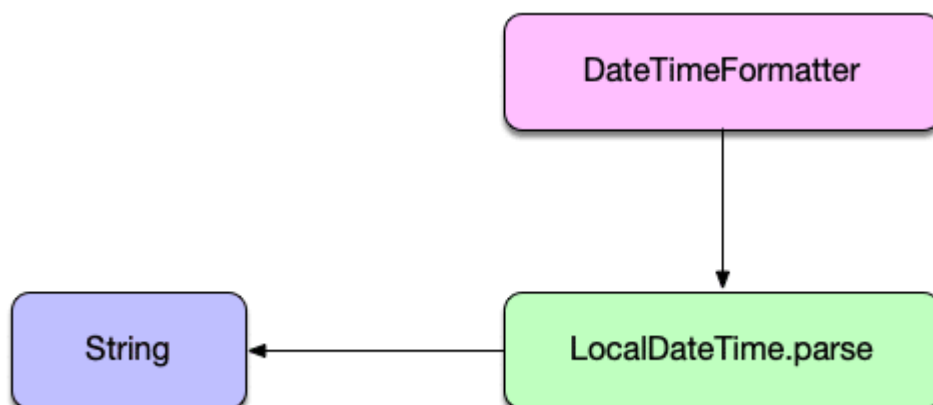
}
```

El resultado lo podemos ver por la consola:

```
<terminated> Principal2 (1) [Java Applic  
11  
20  
15  
1  
6  
2020
```

Java String to Date CustomFormat

Acabamos de ver una introducción al formato de fechas con Java 8 . En muchas ocasiones no nos valdrá con el formato por defecto sino que necesitaremos personalizarlo a través de un `DateTimeFormatter` .

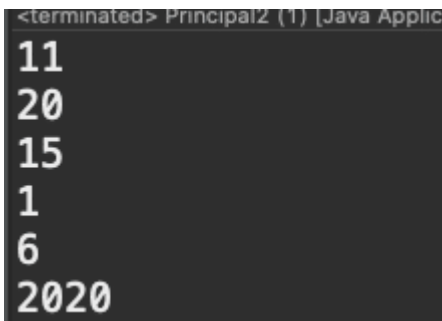


Vamos a ver cómo podemos realizar esta operación.

```
package com.arquitecturajava;  
  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
  
public class Principal2 {  
  
    public static void main(String[] args) {  
        DateTimeFormatter formatter =  
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
```

```
        LocalDateTime localDateTime =  
LocalDateTime.parse("2020-06-01 11:20", formatter);  
        System.out.println(localDateTime.getHour());  
        System.out.println(localDateTime.getMinute());  
        System.out.println(localDateTime.getSecond());  
        System.out.println(localDateTime.getDayOfMonth());  
        System.out.println(localDateTime.getMonthValue());  
        System.out.println(localDateTime.getYear());  
    }  
}
```

El resultado es identico pero hemos personalizado el formateo decidiendo donde ponemos cada valor y que estructura usamos para separarlo:



```
<terminated> Principal2 (1) [Java Applic  
11  
20  
15  
1  
6  
2020
```

Otros artículos relacionados:

- [Java 8 StringJoiner y manejo de cadenas](#)
- [File to String Java 8 y manejo de ficheros](#)
- [Java 8 Files Walk y Recursividad](#)