

Programação Funcional - Java 8

Paradigmas de Linguagens de Programação

Victor Medeiros

Date Time Java 8

- ❖ Nova API de data e hora - JSR 310;
- ❖ Cobre datas, horas, instantes, períodos, durações;
- ❖ Traz 80% + de Joda-Time para o JDK;
- ❖ Corrige os erros no Joda-Time.



Porque mudar?

Por diversas complicações. Entre elas;

Classes Date e Calendar são mutáveis, possuem muitas limitações, design estranho, bugs, dificuldade de se trabalhar com elas.



Qual a diferença?

- ❖ Imutável;
- ❖ Modelo rico, compreensível e fluente;
- ❖ Separação de cronologias;
- ❖ Rico modelo de dados.

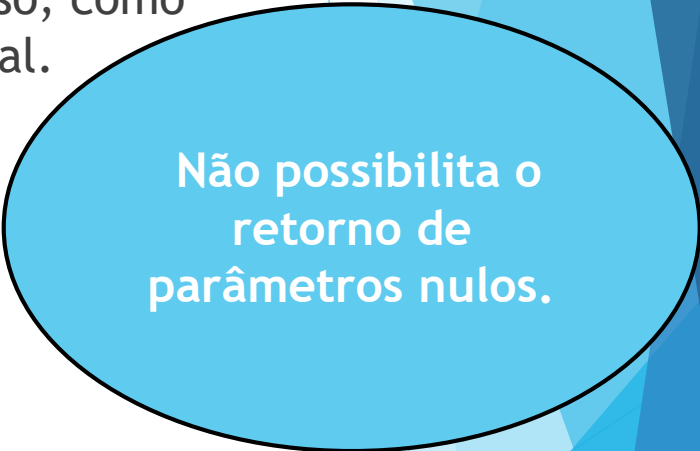


Na prática

Aplicar uma transformação em um Calendar é um processo muito verboso, como por exemplo para criar uma data comum mês a partir da data atual.

```
// incrementando um mês com Calendar  
Calendar mesQueVem = Calendar.getInstance();  
mesQueVem.add(Calendar.MONTH, 1);
```

```
// incrementando um mês com LocalDate  
LocalDate mesQueVem2 = LocalDate.now().plusMonths(1);
```



Não possibilita o
retorno de
parâmetros nulos.

Data

LocalDate(Sem hora)

Datas consistem em dia, mês e ano

Data de nascimento, feriado, apresentação...

```
LocalDate.of(2017, Fri.MAY, 05);
```

Antes

```
public static void main(String[] args) {  
    Calendar c = Calendar.getInstance();  
    c.set(2017, Calendar.MAY, 05);  
    Date data = c.getTime();  
  
    //Data atual  
    DateFormat dataAtual = DateFormat.getDateInstance();  
    System.out.println("Data atual com formatação: " + dataAtual.format(data));  
}
```

Agora

```
public static void main(String[] args) {  
    //Data atual  
    LocalDate data = LocalDate.now();  
    System.out.println("Data: " + data);  
}
```


Hora

LocalTime (Sem Data)

Horários consistem em hora, minutos e segundos ○

Horário de início da aula, horário do despertador

```
LocalTime.of(8, 45);
```

Antes

```
public static void main(String[] args) {  
    Calendar c = Calendar.getInstance();  
    c.set(2017, Calendar.MAY, 05);  
    Date data = c.getTime();  
  
    //Hora atual  
    DateFormat hora = DateFormat.getTimeInstance();  
    System.out.println("Hora formatada: " + hora.format(data));  
}
```

Agora

```
public static void main(String[] args) {  
    //Hora atual  
    LocalTime hora = LocalTime.now();  
    System.out.println("Hora: " + hora);  
}
```

Data e Hora

LocalDateTime

Composto por data e horas

Data e hora de um atendimento, da apresentação de PLP...

```
LocalDateTime.of(2017, Thurth.MAY, 04, 8, 50);
```

Antes

```
public static void main(String[] args) {  
    Calendar c = Calendar.getInstance();  
    c.set(2017, Calendar.MAY, 05);  
    Date data = c.getTime();  
  
    //Data e Hora atuais  
    DateFormat dataHora = DateFormat.getDateTimeInstance();  
    System.out.println(dataHora.format(data));  
}
```

Agora

```
public class DateTime {  
    public static void main(String[] args) {  
        //Data e hora atuais  
        LocalDateTime horaLocal = LocalDateTime.now();  
        System.out.println("Data e Hora: " + horaLocal);  
    }  
}
```

Class	Date	Time	ZoneOffset	ZoneId	Example
LocalDate	✓	✗	✗	✗	2015-12-03
LocalTime	✗	✓	✗	✗	11:30
LocalDateTime	✓	✓	✗	✗	2015-12-03T11:30
OffsetDateTime	✓	✓	✓	✗	2015-12-03T11:30+01:00
ZonedDateTime	✓	✓	✓	✓	2015-12-03T11:30+01:00 [Europe/London]
Instant	✗	✗	✗	✗	123456789 nanos from 1970-01-01T00:00Z

Outros Modelos

Exemplos Completos

<https://github.com/VMedeiros/javaFuncionalPLP>