Seção 11: Tópicos Especiais em Java: data-hora

Alguns Conceitos

- **Data e Hora Local**: Armazenada sem fuso horário.
- **Data e Hora Global**: Armazenada com fuso horário.
- **Duração**: Representa o tempo decorrido entre dois instantes.

Utilidade:

- A **data e hora local** é útil quando o momento exato não precisa ser interpretado em diferentes fusos horários.
  - Exemplo: **Data de nascimento**, **registro de vendas locais**.
- A **data e hora global** é essencial quando o momento exato precisa ser ajustado para diferentes fusos horários.
  - Exemplo: **Transmissões ao vivo**, **agendamento de eventos**, **sistemas multi-região**.

Exemplos:

- **Data e Hora Global**: 2022-07-23T14:30:00Z
  - O sufixo **"Z"** indica o fuso horário de Londres.
  - O sistema deve converter esse horário para o fuso do usuário final.
- **Conversão de Fuso Horário**
  - Se o horário UTC é **14:30**, então:
    - Em Portugal (**UTC+1**) → **15:30**
    - No Brasil (**UTC-3**) → **11:30**

Timezone (fuso horário)

- **GMT (Greenwich Mean Time)**: Horário de referência mundial baseado no meridiano de Greenwich, em Londres.
- **UTC (Coordinated Universal Time)**: Padrão internacional de tempo, muito similar ao GMT, mas com ajustes mais precisos.
- **Z-Time ou Zulu Time**: Outro nome para UTC, indicado pela letra **"Z"** no final de datas e horários.
  - Exemplo: `2022-07-23T14:30:00Z` → Significa **14:30 UTC (horário de Londres).**

Fusos Horários ao Redor do Mundo

Os horários são expressos em relação ao GMT/UTC, por exemplo:

- **São Paulo**: **GMT-3** → 3 horas atrás de Londres.
  - Se em Londres for **10:00**, em São Paulo será **07:00**.
- **Manaus**: **GMT-4** → 4 horas atrás de Londres.
  - Se em Londres for **10:00**, em Manaus será **06:00**.
- **Portugal**: **GMT+1** → 1 hora à frente de Londres.
  - Se em Londres for **10:00**, em Lisboa será **11:00**.

Fusos horários em sistemas

Sistemas e linguagens de programação geralmente utilizam nomes específicos para identificar os fusos, como:

- **"America/Sao_Paulo"** → Para São Paulo.
- **"Europe/Lisbon"** → Para Portugal.
- **"America/New_York"** → Para Nova York.
- **"Asia/Tokyo"** → Para Tóquio.

Padrão ISO 8601 para Datas e Horas

- O **ISO 8601** é um padrão internacional que define como representar **datas e horários** no formato de texto. Esse padrão é amplamente utilizado em **bancos de dados, APIs, sistemas web** e outras aplicações que lidam com datas e tempos.

Formato de Data (sem horário)

Apenas a data (sem hora) é representada no formato:

```
YYYY-MM-DD
```

Exemplo:

- `2024-02-19` → Representa **19 de fevereiro de 2024**

Formato de Data e Hora Local (sem fuso horário)

Se precisar incluir **horário**, o formato segue este padrão:

```
YYYY-MM-DDTHH:MM
YYYY-MM-DDTHH:MM:SS
YYYY-MM-DDTHH:MM:SS.sss
```

Exemplos:

- `2024-02-19T14:30` → **19/02/2024 às 14:30**
- `2024-02-19T14:30:45` → **19/02/2024 às 14:30:45**
- `2024-02-19T14:30:45.123` → **19/02/2024 às 14:30:45 e 123 milissegundos**

Obs: A letra **"T"** separa a **data** do **horário**.

Formato de Data e Hora Global (com fuso horário)

Se precisar indicar **fuso horário**, há duas formas principais:

- **Usando a Letra "Z" (UTC/GMT)**
  - Quando o horário está em **UTC (Greenwich Mean Time)**, adicionamos a letra `Z` no final.

```
YYYY-MM-DDTHH:MM:SSZ
```

Exemplo:

`2024-02-19T14:30:00Z` → **Horário exato UTC (Zulu Time)**

- **Especificando o Deslocamento do Fuso Horário**
  - Se o horário for de outro fuso, indicamos o deslocamento em relação ao UTC:

Exemplos:

`2024-02-19T14:30:00-03:00` → **Horário de São Paulo (UTC-3)**

`2024-02-19T14:30:00+01:00` → **Horário de Lisboa (UTC+1)**

Obs**:** O sinal `+` significa que o horário está **adiantado** em relação ao UTC, e o sinal `-` indica que está **atrasado**.

Operações importantes com data-hora

- **Instanciação**
  - (agora) ⟶ Data-hora
  - Texto ISO 8601 ⟶ Data-hora
  - Texto formato customizado ⟶ Data-hora
  - dia, mês, ano, [horário] ⟶ Data-hora local

- **Formatação**
  • Data-hora ⟶ Texto ISO 8601
  • Data-hora ⟶ Texto formato customizado
- Converter data-hora global para local
  - Data-hora global, timezone (sistema local) ⟶ Data-hora local
  - Obter dados de uma data-hora local
  - Data-hora local ⟶ dia, mês, ano, horário
- Cálculos com data-hora
  - Data-hora +/- tempo ⟶ Data-hora
  - Data-hora 1, Data-hora 2 ⟶ Duração

Pricipais tipos Java 8+

Data-hora local:

- **LocalDate:** Traz apenas a informação data.
- **LocalDateTime:** Além da data, adiciona as informações do tempo.
- **Instant:** Armazena data e a hora no formato GMT/UTC.

Exemplos:

```
// https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/format/DateTimeFormatter.html
DateTimeFormatter fmt1 = DateTimeFormatter.ofPattern("dd/MM/yyyy");
// Indica o formato da data e hora, utilizando a classe DateTimeFormatter
DateTimeFormatter fmt2 = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm");
DateTimeFormatter fmt3 = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss").withZone(ZoneId.systemDefault());

LocalDate d01 = LocalDate.now();
LocalDateTime d02 = LocalDateTime.now();
Instant d03 = Instant.now();

LocalDate d04 = LocalDate.parse("2022-07-20");
LocalDateTime d05 = LocalDateTime.parse("2022-07-20T01:30:26");
// O Instante não tem a função .format()
Instant d06 = Instant.parse("2022-07-20T20:30:26Z");
Instant d07 = Instant.parse("2022-07-20T01:30:26Z");


// https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/time/format/DateTimeFormatter.html
LocalDate d08 = LocalDate.parse("20/07/2022", fmt1);
//or
// LocalDate d08 = LocalDate.parse("20/07/2022", DateTimeFormatter.ofPattern("dd/MM/yyyy"));
LocalDateTime d09 = LocalDateTime.parse("20/07/2022 01:30", fmt2);

LocalDate d10 = LocalDate.of(2022, 07, 20);
LocalDateTime d11 = LocalDateTime.of(2022, 07, 20, 1, 30);

LocalDate r1 = LocalDate.ofInstant(d07, ZoneId.systemDefault());
LocalDateTime r2 = LocalDateTime.ofInstant(d07, ZoneId.of("Asia/Hong_Kong"));

// Calculos com datas e horas
LocalDate pastWeek = d04.minusWeeks(1);
LocalDate plusWeek = d04.plusWeeks(1);
LocalDate minusYear = d04.minusYears(1);
```

```
LocalDate plusMonth = d04.plusMonths(1);
LocalDateTime plusFourHours = d05.plusHours(4);
Instant pastFiveHours = d06.minusSeconds(18000);

// Calculo de diferença entre datas e horas
Duration t1 = Duration.between(d05, plusFourHours);
// Wrong way to calculate the difference between two dates
// Duration t2 = Duration.between(pastWeek, d04);
Duration t2 = Duration.between(pastWeek.atStartOfDay(), d04.atStartOfDay());

System.out.println("d01 = " + d01.toString()); // O toString() é opcional
// 2025-02-25 (Data atual no formato padrão ISO)
System.out.println("d02 = " + d02);
// 2025-02-25T15:17:01.896353345 (Data e hora atual no formato ISO)
System.out.println("d03 = " + d03.toString());
// 2025-02-25T18:17:01.896385062Z (Momento atual em UTC/GMT)
System.out.println("d04 = " + d04);
// 2022-07-20 (Data fixa no formato ISO)
System.out.println("d05 = " + d05);
// 2022-07-20T01:30:26 (Data e hora fixa no formato ISO)
System.out.println("d06 = " + fmt3.format(d06));
// 2022-07-20T01:30:26Z (Momento atual de acordo com a localidade do sistema)
System.out.println("d07 = " + d07);
// 2022-07-20T04:30:26Z (Ajustado para UTC a partir de -03:00, somando 3 horas)
System.out.println("d08 = " + d08);
// 2022-07-20 (Data formatada com padrão dd/MM/yyyy e convertida para ISO)
System.out.println("d09 = " + d09);
// 2022-07-20T01:30 (Data e hora formatadas e convertidas para ISO)
System.out.println("d10 = " + d10);
// 2022-07-20 (Criado com valores individuais de ano, mês e dia)
System.out.println("d11 = " + d11);
// 2022-07-20T01:30 (Criado com valores individuais de ano, mês, dia, hora e minuto)
System.out.println("r1 = " + r1);
// 2022-07-19 (Data convertida para a localidade do sistema, por isso a diferença de -1 dia)
System.out.println("r2 = " + r2);
// 2022-07-20T09:30 (Data convertida para a localidade de Hong Kong +8)
System.out.println("d09 horário: " + d09.getHour());
// 1 (Hora do LocalDateTime)
System.out.println("1 semana atrás: " + pastWeek);
// 2022-07-13 (Data de 1 semana atrás)
System.out.println("1 semana depois: " + plusWeek);
// 2022-07-27 (Data de 1 semana depois)
System.out.println("1 ano atrás: " + minusYear);
// 2021-07-20 (Data de 1 ano atrás)
System.out.println("1 mês depois: " + plusMonth);
// 2022-08-20 (Data de 1 mês depois)
System.out.println("4 horas depois: " + plusFourHours);
// 2022-07-20T05:30:26 (Data e hora de 4 horas depois)
System.out.println("5 horas atrás: " + pastFiveHours);
// 2022-07-20T00:30:26Z (Momento de 5 horas atrás)
//System.out.println("Diferença entre d05 e d06: " + t1.getSeconds() / 60 / 60 + " horas");
System.out.println("Diferença entre d05 e 4 horas depois: " + t1.toHours() + " horas");
// 4 horas (Diferença entre d05 e 4 horas depois)
System.out.print("Diferença entre 1 semana atrás e hoje: " + t2.toDays() + " dias");
// 7 dias (Diferença entre 1 semana atrás e hoje)
```

Listar todos os Zone Id’s

```
for (String s : ZoneId.getAvailableZoneIds()) {
    System.out.println(s);
}
```