Unicesumar

Programação I

Classe LocalDate

Prof. Me. Fábio Perfetto

LocalData

Em diversas aplicações é necessário manipular datas de alguma forma, seja para salvar a data de algum evento que aconteceu, como a data de nascimento de uma pessoa, a data de admissão de uma pessoa em uma empresa ou de uma festa. Algumas das possíveis operações de datas são: comparação entre elas, criação de um filtro para encontrar um evento que aconteceu entre duas datas e o cálculo do número de dias ou meses que faltam para um evento

A classe principal dessa especificação é a classe java.time.LocalDate, que tem a mesma função que a antiga classe **java.util.Date**, mas com muito mais funcionalidades implementadas.

Exemplo de Uso

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

System.out.println(localDate);

System.out.println("Dia da semana: " +
localDate.getDayOfWeek().name());

System.out.println("Dia da semana: " +
localDate.getDayOfWeek().ordinal());

System.out.println("Mes: " + localDate.getMonthValue());

System.out.println("Mes: " + localDate.getMonth().name());

System.out.println("Ano: " + localDate.getYear());
```

Resultado

2016-02-06

Dia da semana: SATURDAY

Dia da semana: 5

Mes: 2

Mes: FEBRUARY

Ano: 2016

Comparação entre datas

Outra coisa bastante utilizada quando trabalhamos com datas é a comparação entre elas como, por exemplo, se uma data é antes ou depois da outra ou quantos meses de diferença existem entre duas datas.

As principais maneiras de comparar datas com os métodos isAfter, isBefore e isEqual.

Veja que a própria classe LocalData tem alguns métodos para a comparação de duas datas, como os métodos isAfter e isBefore, que verificam se uma data é antes ou depois de outra, respectivamente. Para descobrir o tempo passado entre uma data e outra é utilizada a classe Period, onde é possível recuperar a diferença de anos, dias e meses.

Exemplo

```
LocalDate localDateAntigo = LocalDate.of(2010, 3, 7);
LocalDate localDateNovo = LocalDate.of(2015, 3,5);

System.out.println(localDateAntigo.isAfter(localDateNovo));
System.out.println(localDateAntigo.isBefore(localDateNovo));
System.out.println(localDateAntigo.isEqual(localDateNovo));

Period periodo = Period.between(localDateAntigo, localDateNovo);
System.out.println(periodo.getYears() + " Anos " +
periodo.getMonths() + " Meses " + periodo.getDays() + " Dias");
System.out.println("Apenas meses: " + periodo.toTotalMonths());
```

Usando o formatador para converter para Date

Para converter de String para Date utilizaremos o método parse() do objeto formato:

```
Date dataFormatada = formato.parse(dataRecebida);
```

Ex:

```
SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy");
Date data = formato.parse("23/11/2015");
```

Formatador / ou -

A formatação de data para diferentes padrões também ficou um pouco mais simples nessa nova versão, para isso, agora é criado um formatador de dado com a classe DateTimeFormatter e a própria classe LocalDate tem um método format que retorna uma String com a data formatada no padrão passado como parâmetro.

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

DateTimeFormatter formatadorBarra = DateTimeFormatter.ofPattern("dd/MM/yyyy");

DateTimeFormatter formatadorTraco = DateTimeFormatter.ofPattern("dd-MM-yyyy");

System.out.println("Data com /: " + hoje.format(formatadorBarra));

System.out.println("Data com -: " + hoje.format(formatadorTraco));
```

RESULTADO:

Data com /: 06/02/2016

Data com -: 06-02-2016

Outras funcionalidades

Os exemplos anteriores cobriram boa parte da nova API de datas do Java, porém existem muitas outras possibilidades, a API é bastante completa e facilita muito a manipulação de datas. Apenas como exemplo de mais algumas coisas que podem ser feitas mostramos alguns métodos interessantes como, por exemplo, o que verifica se o ano da data é bissexto, o número de dias do mês e do ano, e também a maior e menor data possível na API.

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


System.out.println("Ano bissexto: " + data.isLeapYear());
System.out.println("Número de dias do mês: " + data.lengthOfMonth());
System.out.println("Número de dias do ano: " + data.lengthOfYear());
System.out.println("Menor data: " + LocalDate.MIN);
System.out.println("Maior data: " + LocalDate.MAX);
```

Resultado

Ano bissexto: true

Número de dias do mês: 29

Número de dias do ano: 366

Menor data: -999999999-01-01

Maior data: +999999999-12-31


Unicesumar