

Introdução

Transcrição

Olá! Bem-vindo ao curso de Java Brasil. Veremos como trabalhar com as peculiaridades do sistema no Brasil, de acordo com os documentos que temos aqui. Alguns deles são CPF, CNPJ e título de eleitor. Veremos como validar e formatar esses dados.

Veremos também como trabalhar com dinheiro, ou seja, o Real. Além disso, estudaremos como buscar um CEP na internet e usar no nosso sistema.

Durante esse treinamento, mostrarei algumas das maneiras para validar esses documentos brasileiros. É um curso introdutório, então veremos o passo a passo da construção dessas validações.

Para isso, usaremos algumas bibliotecas que selecionei. A primeira é a [Stella](http://stella.caelum.com.br/) (<http://stella.caelum.com.br/>), uma biblioteca da Caelum open source, então você pode contribuir caso você queira e usar livremente em suas aplicações. Essa biblioteca nos ajudará a validar o CPF, pois veremos qual o algoritmo por trás da validação nos dois últimos dígitos.

[WIKI](#)
Documentação

[DOWNLOAD](#)
Downloads

[SUPPORT](#)
Listas de discussão

[ISSUES](#)
Issue tracker

 **Stella** *Simplificando o desenvolvimento no Brasil.*

Bean Validation

```
@Entity
public class Modelo {
    @Id
    @GeneratedValue
    private Long id;
    @CNPJ
    private String cnpj;
    @CPF
    private String cpf;
}
```

JSF 2

```
<h:outputLabel for="cpfFormatado" value="cpf com formatacao:" />
<h:inputText id="cpfFormatado" value="#{usuarioBean.cpfFormatado}"
    <stella:validateCPF formatted="true" />
</h:inputText>
<!-- outra opção -->
<h:outputLabel value="cpf:" />
<h:inputText id="cpf">
    <f:validator validatorId="StellaCPFValidator" />
</h:inputText>
```

Validadores

```
String cpf = "867.554.707-24";
List<ValidationMessage> messages =
    new CPFValidator().invalidMessagesFor(cpf);
for (ValidationMessage error : messages) {
    System.out.println(error.getMessage());
}
new CPFValidator().assertValid("867.554.707-24");
```

Conversor de número por extenso

```
FormatoDeExtenso formato =
    new FormatoDeSegundosComMilesimos();
NumericToWordsConverter converter =
    new NumericToWordsConverter(formato);
String tempo = converter.toWords(9.52);
String message = "Técnico diz que Bolt poderia " +
    "ter feito 100m em " + tempo;
System.out.println(message);
// Técnico diz que Bolt poderia ter feito 100m
// em nove segundos e quinhentos e vinte milésimos de segundo
```

Abordaremos também a formatação do CPF. Pegaremos a string com o CPF e lidaremos com a formatação com os dois pontos e o hífen. Se você recebeu esse CPF do front-end com a máscara, saberá como tirá-la e trabalhar apenas com os números.

Usaremos também o [Java Money](http://javamoney.github.io/ri.html) (<http://javamoney.github.io/ri.html>), que é uma implementação JSR. A especificação deve entrar no Java 9, e é feita para lidar com dinheiro dentro do Java. Usaremos a Moneta, a implementação oficial dessa JSR.

Moneta

31 May 2015

JSR 354 RI

Reference Implementation of JSR 354

[Download .zip](#) [Download .tar.gz](#) [View on GitHub](#)

Welcome to Moneta - the JSR 354 reference implementation

This module implements JSR 354 Money & Currency. Hereby basically implementations of amounts, currency and roundings are provided. You can access the RI by adding the following Maven dependencies:

```
<dependency>
  <groupId>org.javamoney</groupId>
  <artifactId>moneta</artifactId>
  <version>1.1</version>
</dependency>
```

Também veremos como consumir dados de um CEP, o puxando da internet e colocando em nossa aplicação. Para isso usaremos o [Via CEP \(http://viacep.com.br/\)](http://viacep.com.br/).



VIA CEP
Consulte CEPs de todo o Brasil

Procurando um [webservice](#) gratuito e de alto desempenho para consultar Códigos de Endereçamento Postal (CEP) do Brasil? Utilize o serviço, melhore a qualidade de suas aplicações web e colabore para manter esta base de dados atualizada.

Acessando o webservice de CEP

Para acessar o webservice, um CEP no formato de **{8}** dígitos deve ser fornecido, por exemplo: "01001000". Após o CEP, deve ser fornecido o tipo de retorno desejado, que deve ser "json", "xml", "piped" ou "query".

Exemplo: viacep.com.br/ws/01001000/json/

Validação do CEP

Quando consultado um CEP de formato inválido, por exemplo: "950100100" (9 dígitos), "95010A10" (alfanumérico), "95010 10" (espaço), o código de retorno da consulta será um **400** (Bad Request). Antes de acessar o webservice, valide o formato do CEP e certifique-se que o mesmo possua **{8}** dígitos. Exemplo de como validar o formato do CEP em javascript está disponível nos exemplos abaixo.

Espero que você goste. Vamos começar o curso?