05

Resumo

Transcrição

Vamos recapitular tudo o que vimos até agora? Começamos conhecendo o algoritmo que valida um CPF, e descobrimos que ele é bem complexo. Envolve duas etapas, uma para cada dígito verificador, ambas com muitas multiplicações. E, para que um CPF seja válido, ambos dígitos verificadores precisam ser.

Para não precisar programar esse algoritmo, usamos o framework Stella, da Caelum. Com ele, fizemos a validação usando um `validador`, que é único para cada documento. Depois refatoramos e criamos um método que recebe o `Validator` e o documento em questão, que pode ser o CNPJ, o CPF ou o título de eleitor. Ele avisa se é válido ou não, e avisa quando não é válido. Para isso, fizemos o `try/catch`, que captura a exceção e mostrar qual é o erro.

Outra coisa interessante é que é possível passar o número com a máscara, ou seja, a formatação que inclui os pontos, traços ou barras. Mas esses caracteres ocupam muito espaço, e se quisermos mandar os CPFs para o banco de dados, esses caracteres ocuparão muito espaço. Outro caso em que esses caracteres nos atrapalham é se for preciso enviar os dados para uma API REST de pagamento que não aceita a máscara.

Por isso, aprendemos também a formatar os dados, usando um formatador. Assim, podemos lidar com o documento com e sem formatação, usando as opções `format` e `unformat`, respectivamente.

Já sabemos a validação e a formatação usando o Stella. Lembro que você pode contribuir para a biblioteca com seu GitHub e usar à vontade. Até a próxima!