03

Primeiros Passos com Programação Funcional

Capítulo 1 - Primeiros passos com programação funcional

Seja bem-vindo ao curso de introdução à programação funcional em *Clojure*! Nosso objetivo aqui é aprender sobre o *pensamento funcional*, mais do que essa linguagem que já é bem famosa.

Por pensamento funcional queremos dizer que mudaremos nossa maneira de programar. Não mais de maneira imperativa, ou seja, programar linha após linha, comando após comando expressando ordens. Programaremos de uma maneira muito mais otimizada, por isso *funcional*.

Só para já termos uma ideia do que estamos falando, uma mudança de paradigma na programação funcional é que as variáveis não variam! Este e outros casos nos dará uma visão diferente de como escrever um bom código, como lidar com métodos, funções, algoritmos, etc.

Veremos que muitos programas escritos em uma linguagem imperativa são enormes e numa funcional os mesmos são muito menores. Como exemplo, implementaremos um jogo de forca, o qual em *C* gasta quase 200 linhas. Aqui, com quarenta linhas de código, teremos o mesmo resultado.

A linguagem *Clojure*

Como falamos acima, o que nos interessa aqui é entender o pensamento funcional mais do que aprender a programar em *Clojure*, uma linguagem que dá suporte a isso. Usaremos o *Leiningen*, que facilitará nossa vida na hora de criar um projeto. Em linguagem funcional também temos o costume de experimentar comandos no Terminal, então devemos estar acostumados com ele.

Lá, para habilitar o uso do *Leiningen* digitamos

```
lein repl
```

Se você já programou em outra linguagens, deve estar acostumado com suas "sujeiras": ;, {, [, etc. No *Clojure* usaremos bastante os parênteses para invocar funções:

```
user=> (+ 1 2)
```

Aqui escrevemos:

- `( )` para indicar função,
- `+` para indicar a função de adição,
- `1 2` para indicar os parâmetros que serão somados.

Ao darmos ENTER:

```
user=> (+ 1 2)
3
```

De fato, a soma é feita. Mais exemplos, agora de multiplicação e divisão:

```
user=> (* 5 4)
20

user=> (/ 6 2)
3
```

E assim por diante.

Não existem variáveis em programação funcional

Agora que já tivemos uma introdução de como funciona a escrita em *Clojure*, vamos falar de mudança de paradigmas. Se estivéssemos programando numa linguagem imperativa, poderíamos ter algo assim:

```
int a = (+ 1 2)
```

Estaríamos guardando um valor em uma variável do tipo inteiro. Porém isso não existe aqui, afinal na linguagem funcional não há variáveis! Parece estranho mas veja um exemplo onde definimos uma idade:

```
user=> (def idade 29)
#'user/idade
```

Podemos fazer operações:

```
user=> (+ idade 3)
32
```

Porém a "variável" `idade` nunca muda:

```
user=> idade
29
```

Veremos como trabalhar com essa ideia, que a princípio parece sem cabimento, mais para frente. Porém, agora vamos começar criando um projeto.

Nosso projeto em linguagem funcional

Nosso projeto para este curso é desenvolver um jogo de forca. Para isso vamos começar do início e sair do *Leiningen* digitando `exit` . Vamos criar uma aplicação em *Clojure*:

```
lein new app forca
Generating a project called forca based on the 'app' template.
```

Ou seja, gerou-se um projeto chamado "forca" com base no *template* "app". Foi criado um diretório para a aplicação com uma estrutura mínima padrão. Para entrarmos nele e, respectivamente, visualizar seus arquivos, fazemos:

```
cd forca/

ls
```

Abrimos um editor de texto. Nós do Alura gostamos bastante do Sublime, então ele será usado como referência. O arquivo que nos interessa aqui encontra-se em "/forca/src/forca" e se chama "core.clj". Será nesse arquivo que escreveremos o código. Por padrão ele já vem com:

```
(ns forca.core
  (:gen-class))

(defn -main
  "I don't do a whole lot ... yet."
  [& args]
  (println "Hello, World"))
```

Existe um método *main* com exemplo de código.

Comecemos, então, a implementar nosso jogo de forca! A primeira coisa será definir o total de vidas de um jogador. Sabemos que toda vez que ele erra, perde uma vida:

```
(ns forca.core
  (:gen-class))

(def total-de-vidas 6)
```

No terminal, se digitarmos `total-de-vidas` , ele retornará "6".

Esse é o começo do nosso aprendizado. A principal lição foi que **VARIÁVEIS NÃO VARIAM** e será dessa forma que iremos programar.