05. Distribuindo comandos e tratamento de erro

Date	@08/09/2022
Categoria	Java
Curso	Threads em Java 2: programação concorrente avançada

Tópicos

Revisão (Thread 2)

- Thread e Runnable
- ExecutorService - Pool de Threads
  - CachedThreadPool
  - FixedThreadPool
- thread.join()
- volatile e AtomicBoolean

O que aprendemos?

- Cada thread possui a sua pilha de métodos.
- O tratamento de exceções deve ser específico para cada pilha.
- Podemos plugar um `UncaughtExceptionHandler` para centralizar o tratamento.
- O pool de threads oferece uma fábrica de threads para personalizar a criação da thread.

Um pouco sobre padrões de projetos

Uma vez um desenvolvedor experiente falou para mim:

Padrões de projeto não foram inventados, eles vem da prática e alguém deu um nome bonito para tal!

Pois é, de vez em quando usamos um padrão de projeto sem saber que ele existe! No nosso projeto já aplicamos alguns, principalmente no lado do servidor.

Talvez o padrão mais simples de enxergar o **Factory Method** que utilizamos dentro da nossa fábrica de threads (`FabricaDeThreads`). Aquele único método `newThread(..)` é um *factory method* que encapsula a criação de uma thread. O pool de threads usa aquela fábrica para criar uma thread, usa o *Factory Method*.

Outro padrão interessante aplicamos na classe `DistribuirTarefas`. Apesar da implementação simples, ela segue um padrão chamado de **FrontController**. Esse

padrão na verdade vem do mundo de desenvolvimento web e representa uma entrada única na aplicação. Na nossa aplicação todos os "pedidos" dos clientes passam pela classe `DistribuirTarefas`. Ela centraliza o fluxo, analisa o pedido e decide (controla) o que é para executar. Isso é o papel do controlador ou *FrontController*.

Por fim, temos os nossos comandos que seguem o padrão **Command**.
Um *Command* encapsula a execução de "algo", encapsula alguma ação ou lógica. Em alguns casos os comandos são chamados de *actions* e eles realmente possuem todo o código para atender aquele pedido especifico do cliente. Enquanto o controlador analisa o pedido e decide qual comando a usar, o *Command* realmente possui a lógica.

Bem vindo ao mundo fantástico dos padrões de projetos :)

Lista de padrões de projeto

- [ ] Factory Method
- [ ] FrontController
- [ ] Command