12

Para saber mais: FutureTask

Temos duas interfaces para definir a tarefa de uma thread: a antiga `Runnable` e a mais recente `Callable`.

A diferença é que `Callable` possui um retorno e trabalha com exceções do tipo *checked*. Além disso, um `Callable` só pode ser executado através de um *pool de threads*.

Como já falamos, não se pode usar uma thread com um `Callable`! Veja o código abaixo que **não compila** pois o construtor da classe `Thread` só funciona com `Runnable`:

```
Callable<String> tarefa = new Tarefa(); //Tarefa implementa Callable
new Thread(tarefa).start(); //não compila!!
```

Isso significa que somos obrigados de usar um pool com `Callable`? E se no meu código não houver a possibilidade de usar um pool de threads?

Felizmente, há uma solução para resolver esse impasse que se chama [FutureTask (https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/FutureTask.html)](https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/FutureTask.html)!

Veja o código:

```
Callable<String> tarefa = new Tarefa(); //Tarefa implementa Callable
FutureTask<String> futureTask = new FutureTask<String>(tarefa);
new Thread(futureTask).start(); //usando Thread puro!!
String resultado = futureTask.get();
```

Isso funciona pois o `FutureTask` também é um `Runnable` (implementa a interface `Runnable`)! Repare também que o `FutureTask` recebe no construtor o `Callable` (a tarefa).

Podemos concluir que o `FutureTask` é um intermediário entre `Callable` e `Runnable`.