# fibers (fibras)

**Fibers** são um recurso introduzido no PHP 8.1 que permite criar funções que podem ser **pausadas** e **retomadas** de maneira mais poderosa e flexível do que os **Generators**. Elas fornecem um controle total sobre a execução do código, permitindo que você pause uma execução em qualquer ponto da pilha de chamadas e a retome posteriormente.

* **Fibers são Classes**: No PHP, as Fibers são representadas pela classe Fiber. Você pode criar e manipular uma Fiber usando esta classe.

## Propriedades e Métodos da Classe Fiber

A classe Fiber possui propriedades e métodos que facilitam a criação, suspensão e retomada de Fibers.

### A. Métodos da Classe Fiber

1. **Fiber::\_\_construct(callable $callback)**
   * **Descrição**: Cria uma nova Fiber. O $callback é a função que a Fiber executará quando for iniciada.
   * **Uso**: Você passa uma função anônima ou callback para ser executada dentro da Fiber.
2. **Fiber::start(mixed ...$args): mixed**
   * **Descrição**: Inicia a execução da Fiber com os argumentos fornecidos. Retorna o valor da primeira chamada a Fiber::suspend(), ou o valor retornado pelo callback se ele não for suspenso.
   * **Uso**: Você chama este método para começar a executar a Fiber.
3. **Fiber::suspend(mixed $value = null): mixed**
   * **Descrição**: Suspende a execução da Fiber e retorna o controle ao chamador. O $value é passado para Fiber::resume() ou Fiber::throw() quando a Fiber for retomada.
   * **Uso**: Pausa a execução da Fiber em qualquer ponto, aguardando ser retomada.
4. **Fiber::resume(mixed $value = null): mixed**
   * **Descrição**: Retoma a execução da Fiber suspensa, retornando o $value para a chamada Fiber::suspend().
   * **Uso**: Retoma a Fiber e fornece um valor que será retornado pela chamada a Fiber::suspend().
5. **Fiber::throw(Throwable $exception): mixed**
   * **Descrição**: Retoma a execução da Fiber suspensa e lança uma exceção dentro dela. A exceção é capturada dentro da Fiber.
   * **Uso**: Retoma a execução da Fiber lançando uma exceção.
6. **Fiber::isStarted(): bool**
   * **Descrição**: Retorna true se a Fiber tiver sido iniciada.
   * **Uso**: Verifica se a Fiber já foi iniciada.
7. **Fiber::isSuspended(): bool**
   * **Descrição**: Retorna true se a Fiber estiver atualmente suspensa.
   * **Uso**: Verifica se a Fiber está em um estado suspenso.
8. **Fiber::isRunning(): bool**
   * **Descrição**: Retorna true se a Fiber estiver em execução e não estiver suspensa.
   * **Uso**: Verifica se a Fiber está ativa.
9. **Fiber::isTerminated(): bool**
   * **Descrição**: Retorna true se a Fiber tiver terminado (ou seja, se a execução do callback estiver concluída).
   * **Uso**: Verifica se a Fiber terminou.

## Mas pra que realmente servem as fibers ?

As fibers servem para iniciarmos uma função, pausarmos ela, e depois começarmos de novo. Mas fazer isso apenas 1 vez, sem ciclos.

Texto

Descrição gerada automaticamentePodemos ter fibers aninhadas, que são muito mais fácis de montar do que com generators, mas enfim, veremos a sintaxe de uma fiber:

Primeiro montarmos a fiber com os códigos que queremos.

Então, quando mandamos iniciar a fiber com seu método start(), ela irá fazer tudo que esta antes de Fiber::suspend():

Quando chegar Fiber::suspend(), a função será pausada.



Depois irá retornar, e terminar o resto da função, apenas quando usarmos o resume().

## Diferenças Entre Generators e Fibers

Embora Generators e Fibers possam parecer semelhantes, eles têm diferenças fundamentais:

**A. Controle de Execução**

* **Generators**: Pausam a execução apenas nos pontos de yield. Eles são iteradores e devem ser usados em loops.
* **Fibers**: Podem ser pausadas e retomadas em qualquer ponto na pilha de chamadas, oferecendo controle total sobre o fluxo de execução.

**B. Direção de Dados**

* **Generators**: São unidirecionais. Eles produzem valores para serem consumidos pelo código que os chama.
* **Fibers**: São bidirecionais. Elas podem pausar, enviar valores para o chamador e receber valores ou exceções de volta quando retomadas.

**C. Propósito e Uso**

* **Generators**: Usados principalmente para iterar sobre grandes conjuntos de dados ou criar iteradores personalizados.
* **Fibers**: Usadas para multitarefa cooperativa, operações de E/S não bloqueantes ou situações em que você precisa de controle total sobre a execução.