PHP asíncrono: una introducción a las fibras
#php#desarrollador web#programación#codificación
Con la llegada de PHP 8.1, se nos presentó una característica nueva y emocionante que promete transformar la forma en que abordamos la programación asincrónica en PHP: **Fibers** . Es un cambio que no solo simplifica la forma en que escribimos código, sino que también mejora drásticamente la eficiencia de nuestras aplicaciones.
En este artículo, vamos a explorar el mundo de las fibras en PHP. Descubriremos qué son, comprenderemos su importancia y explicaremos cómo usarlos de manera óptima en sus aplicaciones PHP. Tanto si es un desarrollador de PHP experimentado como si acaba de empezar, hay algo que aprender aquí mientras navegamos por las aguas asincrónicas de PHP usando Fibers. Entonces, ¡vamos a sumergirnos!

¿Qué son las fibras?

Primero, obtengamos una comprensión básica de las fibras. En informática, una fibra es un mecanismo que nos permite escribir código asíncrono en un estilo más directo y de apariencia síncrona. PHP 8.1 agregó soporte nativo para fibras, lo que significa que ahora podemos manejar tareas como operaciones de E/S, solicitudes de red o consultas de bases de datos de manera más eficiente, sin bloquear la ejecución del resto de nuestro código.

¿Por qué importan las fibras?

Las fibras son importantes porque hacen que la programación asíncrona en PHP sea mucho más intuitiva. Antes de las fibras, si quería escribir código PHP asíncrono, tenía que confiar en extensiones como Swoole o ReactPHP . Estas son herramientas poderosas, pero requieren que escriba código en un estilo un tanto enrevesado y orientado a la devolución de llamada.
Las fibras, por otro lado, nos permiten escribir código asíncrono que se ve y se siente como código síncrono. Esto hace que nuestro código sea más fácil de leer, escribir y mantener.
Tenga en cuenta que también es posible pausar y reanudar Fibers, que trataremos en una publicación diferente.

Ejemplo básico

En esencia, una fibra es un hilo ligero de ejecución. Esto nos permite escribir código asíncrono sin bloqueo.
He aquí un ejemplo básico:

```
<?php

function syncHttpRequest(string $url) {
    sleep(3);
}

function asyncHttpRequest(string $url): Fiber {
    return new Fiber(function () use ($url): void {
        // Simulating the time for a slow HTTP request
        sleep(3);
    });
}

$urls = [
    'https://example.com/api1',
```

```
    'https://example.com/api2',
    'https://example.com/api3',
    'https://example.com/api4',
    'https://example.com/api5'
];


// Execute requests synchronously (one after the other)
foreach($urls as $url) {
    syncHttpRequest($url);
}
// Takes 15 seconds

// Execute requests asynchronously (all at once) using Fibers
foreach($urls as $url) {
    asyncHttpRequest($url)->start();
}
// Takes just 3 seconds!
```

En el ejemplo proporcionado, estamos tratando con dos funciones distintas: `syncHttpRequest` y `asyncHttpRequest`. Cada función acepta una URL como argumento e imita una solicitud HTTP al iniciar un período de "reposo" durante tres segundos. Lo que lo `asyncHttpRequest` distingue es su capacidad para generar y devolver un nuevo Fiber para cada entrada de URL.

A continuación, construimos una matriz que contiene cinco URL de marcador de posición. Utilizando dos bucles for separados, invocamos secuencialmente `syncHttpRequest` y luego `asyncHttpRequest` para cada URL en la matriz.

En el ciclo que aprovecha `syncHttpRequest()`, encontramos un período de espera de tres segundos entre cada llamada, lo que significa que la próxima llamada no se puede iniciar hasta que se complete la anterior. En consecuencia, este bucle requiere aproximadamente 15 segundos para seguir su curso.

Sin embargo, el escenario cambia drásticamente cuando usamos `asyncHttpRequest()` dentro del bucle. Gracias a Fibers, podemos iniciar todas las solicitudes simultáneamente, ejecutándolas de manera simultánea en segundo plano. Esto permite que el ciclo for complete toda su operación en solo 3 segundos, ¡un marcado contraste con su iteración anterior!

Este es un ejemplo simple, pero ilustra el poder de Fibers: nos permiten escribir código que realiza operaciones potencialmente lentas (como solicitudes HTTP, operaciones de E/S, consultas de bases de datos, etc.) de una manera que no bloquea la ejecución del resto de nuestro código.

Trabajar con fibras: algunos consejos

Si bien las fibras pueden simplificar el código PHP asíncrono, también requieren que piense un poco diferente sobre cómo estructura su código.

. Capturar excepciones: las fibras lanzan una `FiberError` excepción cuando fallan, así que asegúrese de capturar estas excepciones en su código.
. Tenga cuidado con el estado compartido: al igual que con los subprocesos, las fibras pueden generar problemas con el estado compartido, especialmente si no tiene cuidado. Asegúrese de administrar adecuadamente los recursos compartidos para evitar condiciones de carrera.

Conclusión

Las fibras representan un importante paso adelante para PHP, proporcionando a los desarrolladores una poderosa herramienta para escribir código eficiente y sin bloqueos de una manera mucho más intuitiva que las soluciones anteriores. Si bien requieren una forma ligeramente diferente de pensar en su código, los beneficios que ofrecen en términos de eficiencia y claridad del código hacen que valga la pena considerarlos para su próximo proyecto PHP.
Esta publicación solo rascó la superficie de las posibilidades con Fibers. Veremos otras características (como pausar y reanudar la ejecución) en otra publicación.
Con suerte, este artículo le brinda una comprensión más clara de qué son las fibras y cómo usarlas de manera efectiva. ¡Así que adelante, pruebe las fibras y vea lo que pueden hacer por sus aplicaciones PHP!

Apéndice

Consulte el manual de PHP para obtener información detallada sobre las fibras:
Resumen de fibras
La clase de fibra
Cover Photo by Héctor J. Rivas on Unsplash.

Esta publicación fue escrita con la ayuda de ChatGPT (GPT-4)