Sviluppo di applicazioni professionali in PHP 8
(parte I)

Dott. Enrico Zimuel

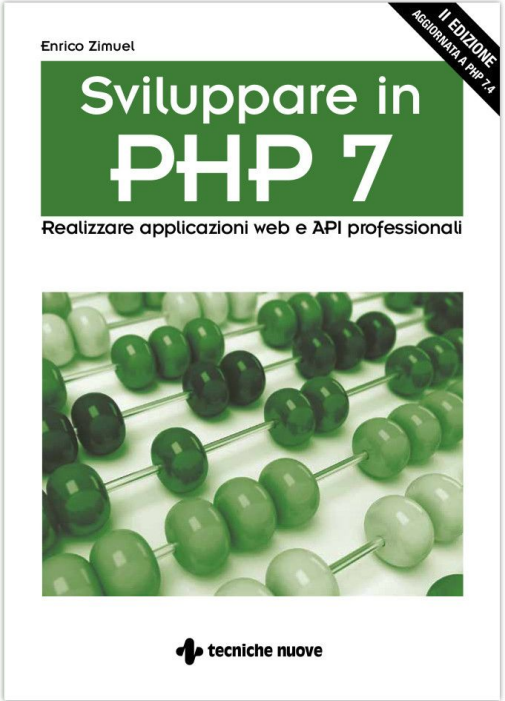
Seminario del 6 Maggio 2022, Eventi CLEII - CLEBA
Università degli Studi "G.D'Annunzio" di Chieti-Pescara

Mi presento

- Programmatore dal 1996
- Principal Software Engineer presso Elastic
- Sviluppatore open source
- PHP Certified Engineer dal 2008
- Autore di libri tecnici (PHP, Javascript, etc)
- Speaker TEDx e relatore in 110+ conferenze
- Professore a contratto Università di Torino e ITS ICT Piemonte
- Laureato in Economia Informatica presso l'Università di Chieti-Pescara
- Ulteriori informazioni: www.zimuel.it

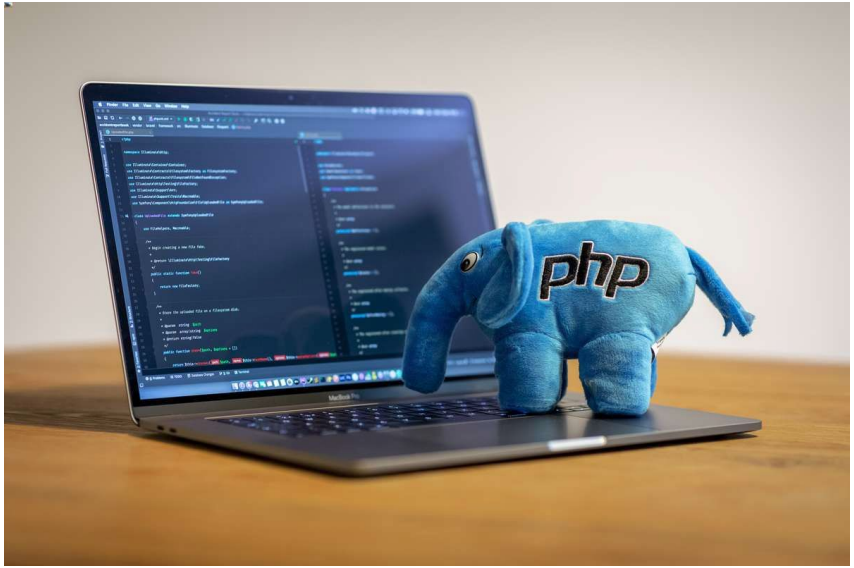
Il mio ultimo libro

- **Sviluppare in PHP 7** (II edizione), realizzare applicazioni web e API professionali
- pp. 432, 2019, Tecniche Nuove editore, ISBN 978-8848140317
- Disponibile anche in formato e-book
- Informazioni: www.sviluppareinphp7.it



Sommario

- Perchè il PHP?
- Popolarità del linguaggio
- Versioni supportate
- Le novità del PHP 8
  - Compilatore JIT
  - Named arguments
  - etc
- Le novità del PHP 8.1
  - Enumerations
  - Fibers e oltre
  - etc



PHP?

- PHP è un linguaggio per lo sviluppo di applicazioni web nato nel 1995
- Ha subito molte trasformazioni negli anni:
  - **PHP 5**, pieno supporto del modello OOP
  - **PHP 7,** performance e tipizzazione parametri in/out
  - **PHP 8**, compilatore JIT, consolidamento, aggiunte funzionalità avanzate
- Il PHP è il linguaggio server-side più utilizzato per applicazioni web (il 77.5% di tutti i siti, dato del 5 Maggio 2022)
- Chi usa il PHP? Wikipedia, Yahoo!, Facebook (Hack fork del PHP), Etsy, Slack, Magento, WordPress, Flickr, iStockPhoto, Baidu, Digg, etc

Popolarità del PHP

PYPL PopularitY of Programming Language

Worldwide, May 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.85 %	-2.5 %
2		Java	17.86 %	-0.1 %
3		JavaScript	9.17 %	+0.4 %
4		C#	7.62 %	+0.7 %
5		C/C++	7.0 %	+0.4 %
6		PHP	5.36 %	-1.0 %
7		R	4.34 %	+0.5 %
8	↑↑↑	TypeScript	2.39 %	+0.7 %

TIOBE index

Apr 2022	Apr 2021	Change	Programming Language	Ratings	Change
1	3	↑	Python	13.92%	+2.88%
2	1	↓	C	12.71%	-1.61%
3	2	↓	Java	10.82%	-0.41%
4	4		C++	8.28%	+1.14%
5	5		C#	6.82%	+1.91%
6	6		Visual Basic	5.40%	+0.85%
7	7		JavaScript	2.41%	-0.03%
8	8		Assembly language	2.35%	+0.03%
9	10	↑	SQL	2.28%	+0.45%
10	9	↓	PHP	1.64%	-0.19%
11	16	⇈	R	1.55%	+0.44%
12	12		Delphi/Object Pascal	1.18%	-0.29%
13	14	↑	Go	1.09%	-0.14%

Versioni supportate del PHP

Branch	Initial Release		Active Support Until		Security Support Until	
7.4	28 Nov 2019	*2 years, 5 months ago*	28 Nov 2021	*5 months ago*	28 Nov 2022	*in 6 months*
8.0	26 Nov 2020	*1 year, 5 months ago*	26 Nov 2022	*in 6 months*	26 Nov 2023	*in 1 year, 6 months*
8.1	25 Nov 2021	*5 months ago*	25 Nov 2023	*in 1 year, 6 months*	25 Nov 2024	*in 2 years, 6 months*

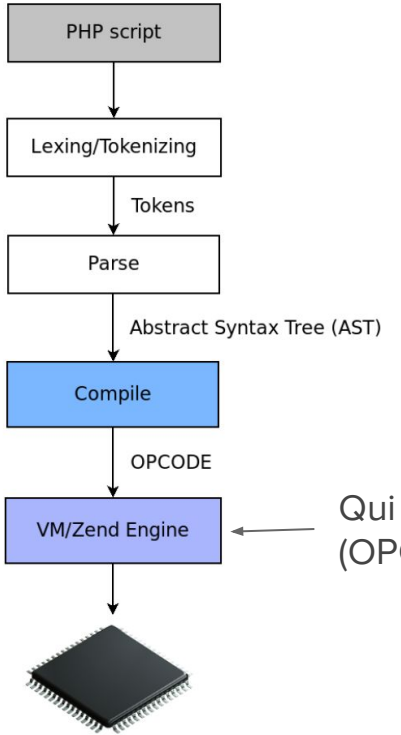
PHP 8

- **PHP 8** è stato rilasciato il 26 Novembre 2020
- Novità principali:
  - Compilatore Just In Time (JIT)
  - Named arguments
  - Attributi
  - Promozione a proprietà degli argomenti del costruttore
  - Tipi unione
  - Espressione match
  - Operatore nullsafe
  - Weak Map

JIT

- Il compilatore **Just In Time** (JIT) è una funzionalità del run-time che consente di tradurre il codice PHP in codice macchina senza dover eseguire il processo di interpretazione del bytecode

Esecuzione del PHP



Qui lo Zend Engine interpreta il bytecode (OPCODE) generato dal compilatore

Esecuzione del PHP

1. Il codice PHP viene tradotto in **bytecode** (OPCODE)
2. Il bytecode viene interpretato (eseguito) dallo **Zend Engine** (la virtual machine del PHP)

Esempio di bytecode

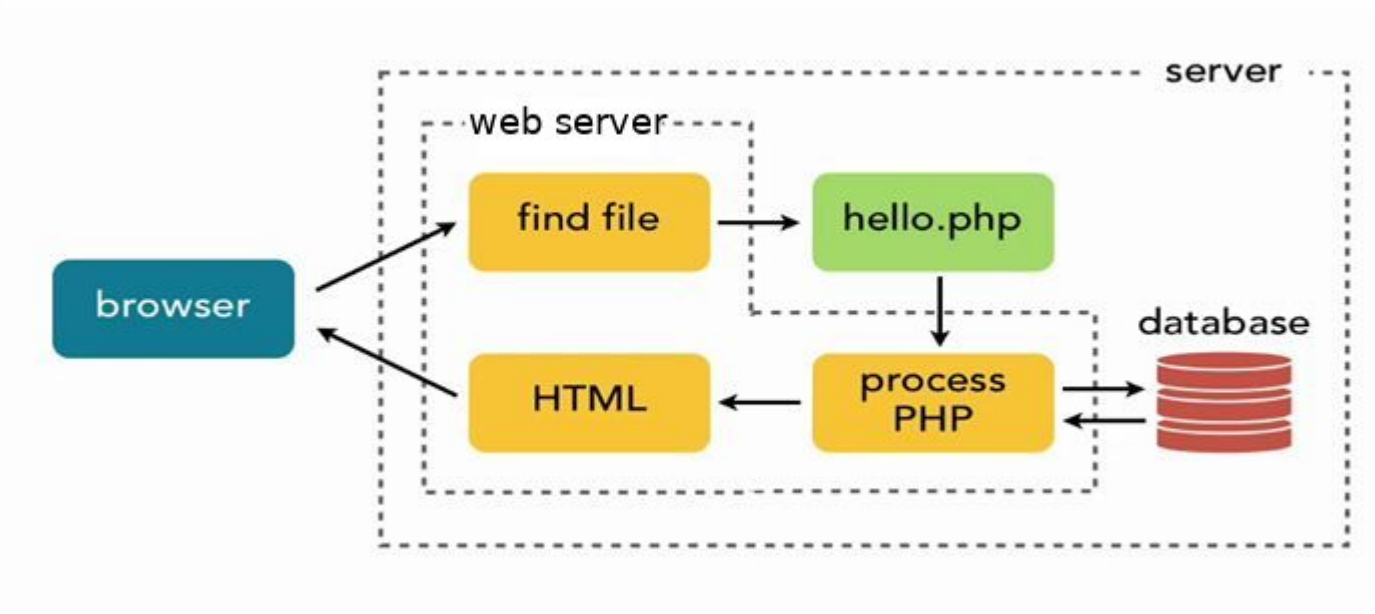
```php
for ($i=0; $i<100; $i++) {
    echo $i;
}
```

Opcode:

```
L0 (2):     ASSIGN CV0($i) int(0)
L1 (2):     JMP L4
L2 (3):     ECHO CV0($i)
L3 (2):     PRE_INC CV0($i)
L4 (2):     T1 = IS_SMALLER CV0($i) int(100)
L5 (2):     JMPNZ T1 L2
L6 (5):     RETURN int(1)
```

php -d opcache.opt_debug_level=0x20000 -d opcache.enable_cli=1 test.php

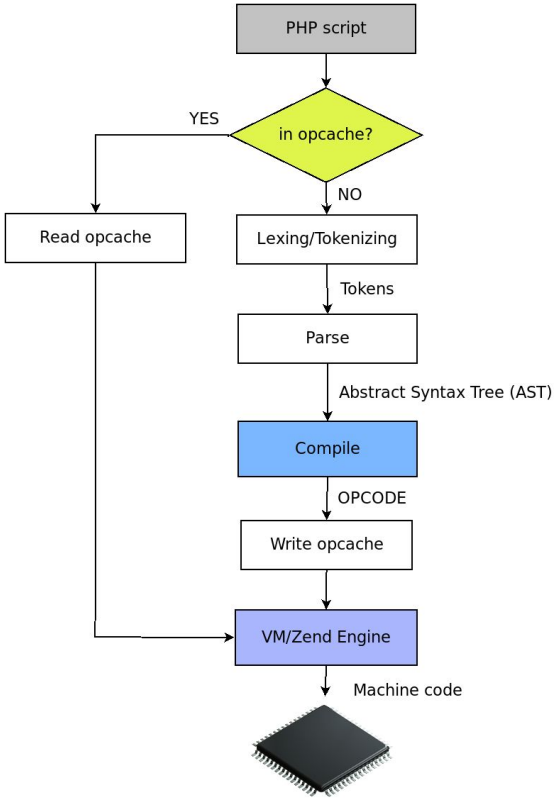
Architettura client/server



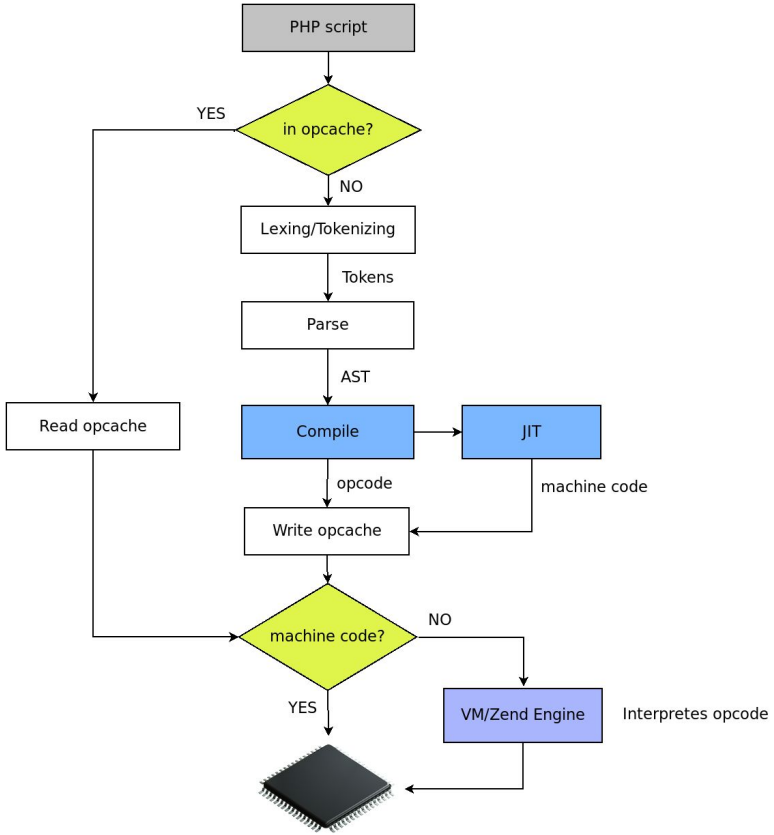
Ottimizzare l'esecuzione del PHP

- Se il sorgente hello.php dell'esempio precedente non cambia tra una richiesta HTTP e l'altra perchè eseguire nuovamente la traduzione in bytecode?
- Il bytecode rimane lo stesso e quindi potrebbe essere messo in una **memoria cache** (**opcache**) per essere riutilizzato da più richieste HTTP diverse

Opcache


PHP script
in opcache?
YES
NO
Read opcache
Lexing/Tokenizing
Tokens
Parse
Abstract Syntax Tree (AST)
Compile
OPCODE
Write opcache
VM/Zend Engine
Machine code

JIT


PHP script
YES
in opcache?
NO
Lexing/Tokenizing
Tokens
Parse
AST
Read opcache
Compile
JIT
opcode
machine code
Write opcache
machine code?
NO
YES
VM/Zend Engine
Interpretes opcode

JIT: codice assembly

```
sub $0x10, %rsp
      lea 0x50(%r14), %rdi
      cmp $0xa, 0x8(%rdi)
      jnz .L1
      mov (%rdi), %rdi
      cmp $0x0, 0x18(%rdi)
      jnz .L6
      add $0x8, %rdi
.L1:
      test $0x1, 0x9(%rdi)
      jnz .L7
      mov $0x0, (%rdi)
      mov $0x4, 0x8(%rdi)
.L2:
      mov $EG(exception), %rax
      cmp $0x0, (%rax)
      jnz JIT$$exception_handler
      jmp .L4
```

```
.L3:
      mov $0x7fcc67e2e630, %r15
      mov $0x561f82773690, %rax
      call *%rax
      mov $EG(exception), %rax
      cmp $0x0, (%rax)
      jnz JIT$$exception_handler
      cmp $0x4, 0x58(%r14)
      jnz .L9
      add $0x1, 0x50(%r14)
.L4:
      mov $EG(vm_interrupt), %rax
      cmp $0x0, (%rax)
      jnz .L11
      cmp $0x4, 0x58(%r14)
      jnz .L12
      cmp $0x64, 0x50(%r14)
      jl .L3
...
```

php -d opcache.enable_cli=1 -d opcache.jit=1235 -d opcache.jit_buffer_size=64M -d opcache.jit_debug=1 test.php

Named arguments

- Prendiamo una funzione del PHP come htmlspecialchars()

```
htmlspecialchars(
    string $string,
    int $flags = ENT_QUOTES | ENT_SUBSTITUTE | ENT_HTML401,
    ?string $encoding = null,
    bool $double_encode = true
): string
```

- Possiamo specificare i parametri tramite il loro nome:

```
htmlspecialchars("<a href='test'>Test</a>", double_encode: false);
```

Attributi

PHP 7

```php
class PostsController
{
    /**
     * @Route("/api/posts/{id}", methods={"GET"})
     */
    public function get($id) { /* ... */ }
}
```

PHP 8

```php
class PostsController
{
    #[Route("/api/posts/{id}", methods: ["GET"])]
    public function get($id) { /* ... */ }
}
```

Promozione a proprietà degli argomenti del costruttore

PHP 7

```php
class Point {
  public float $x;
  public float $y;
  public float $z;

  public function __construct(
    float $x = 0.0,
    float $y = 0.0,
    float $z = 0.0
  ) {
    $this->x = $x;
    $this->y = $y;
    $this->z = $z;
  }
}
```

PHP 8

```php
class Point {
  public function __construct(
    public float $x = 0.0,
    public float $y = 0.0,
    public float $z = 0.0,
  ) {}
}
```

Tipi unione

PHP 7

```php
class Number {
  /** @var int|float */
  private $number;

  /**
   * @param float|int $number
   */
  public function __construct($number) {
    $this->number = $number;
  }
}

new Number('NaN'); // Ok
```

PHP 8

```php
class Number {
  public function __construct(
    private int|float $number
  ) {}
}

new Number('NaN'); // TypeError
```

tipo int o float

Espressione match

PHP 7

```php
switch (8.0) {
  case '8.0':
    $result = "Oh no!";
    break;
  case 8.0:
    $result = "This is what I expected";
    break;
}
echo $result;
//> Oh no!
```

PHP 8

```php
echo match (8.0) {
  '8.0' => "Oh no!",
  8.0 => "This is what I expected",
};
//> This is what I expected
```

Operatore nullsafe

PHP 7

```php
$country =  null;

if ($session !== null) {
  $user = $session->user;

  if ($user !== null) {
    $address = $user->getAddress();

    if ($address !== null) {
      $country = $address->country;
    }
  }
}
```

PHP 8

```php
$country = $session?->user?->getAddress()?->country;
```

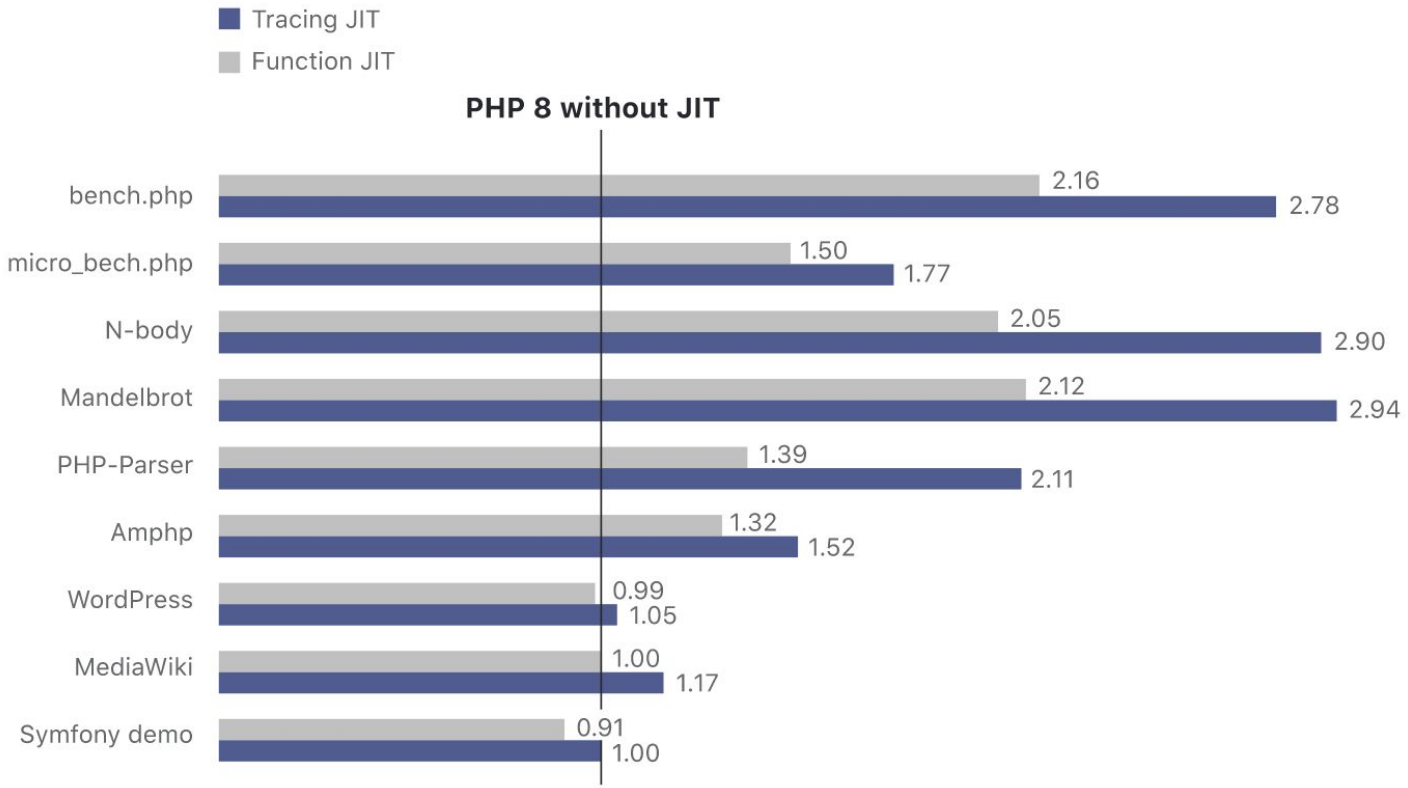
WeakMap

```php
$map = new WeakMap;
$obj = new stdClass;
$map[$obj] = 42;
var_dump($map);
// object(WeakMap)#1 (1) {
//   [0]=> [
//     "key" => object(stdClass)#2,
//     "value" => int(42)
//   ]
// }

unset($obj);
var_dump($map);
// object(WeakMap)#1 (0) { }
```

Esempio: cache con WeakMap

```
class Foo {
    private WeakMap $cache;

    public function getSomethingWithCaching(object $obj) {
        return $this->cache[$obj] ??= $this->somethingExpensive($obj);
    }
}
```

Benchmark di PHP 8



PHP 8.1

- **PHP 8.1** rilasciato il 25 Novembre 2021
- Novità principali:
  - Enumerations
  - Proprietà di sola lettura
  - Sintassi First-class callable
  - New in initializers
  - Intersezione di tipi
  - Never return type
  - Costanti final di classe
  - Fibers

Enumerations

PHP < 8.1

```php
class Status
{
    const DRAFT = 'draft';
    const PUBLISHED = 'published';
    const ARCHIVED = 'archived';
}
function acceptStatus(string $status) {...}
```

PHP 8.1

```php
enum Status
{
    case Draft;
    case Published;
    case Archived;
}
function acceptStatus(Status $status) {...}
```

Proprietà di sola lettura

PHP < 8.1

```php
class BlogData
{
    private Status $status;

    public function __construct(Status $status)
    {
        $this->status = $status;
    }

    public function getStatus(): Status
    {
        return $this->status;
    }
}
```

PHP 8.1

```php
class BlogData
{
    public readonly Status $status;

    public function __construct(Status $status)
    {
        $this->status = $status;
    }
}
```

First-class callable

PHP < 8.1

```php
$foo = [$this, 'foo'];

$fn = Closure::fromCallable('strlen');
```

PHP 8.1

```php
$foo = $this->foo(...);

$fn = strlen(...);
```

New in initializers

PHP < 8.1

```php
class Service
{
    private Logger $logger;

    public function __construct(
        ?Logger $logger = null,
    ) {
        $this->logger = $logger ?? new NullLogger();
    }
}
```

PHP 8.1

```php
class Service
{
    private Logger $logger;

    public function __construct(
        Logger $logger = new NullLogger(),
    ) {
        $this->logger = $logger;
    }
}
```

Intersezione di tipi

PHP < 8.1

```php
function count_and_iterate(Iterator $value) {
    if (!($value instanceof Countable)) {
        throw new TypeError('value must be Countable');
    }

    foreach ($value as $val) {
        echo $val;
    }

    count($value);
}
```

PHP 8.1

```php
function count_and_iterate(Iterator&Countable $value) {
    foreach ($value as $val) {
        echo $val;
    }

    count($value);
}
```

Never return type

PHP < 8.1

```php
function redirect(string $uri) {
    header('Location: ' . $uri);
    exit();
}

function redirectToLoginPage() {
    redirect('/login');
    echo 'Hello'; // <- dead code
}
```

PHP 8.1

```php
function redirect(string $uri): never {
    header('Location: ' . $uri);
    exit();
}

function redirectToLoginPage(): never {
    redirect('/login');
    echo 'Hello'; // <- dead code detected by static analysis
}
```

Costanti final di classe

PHP < 8.1

```php
class Foo
{
    public const XX = "foo";
}

class Bar extends Foo
{
    public const XX = "bar"; // No error
}
```

PHP 8.1

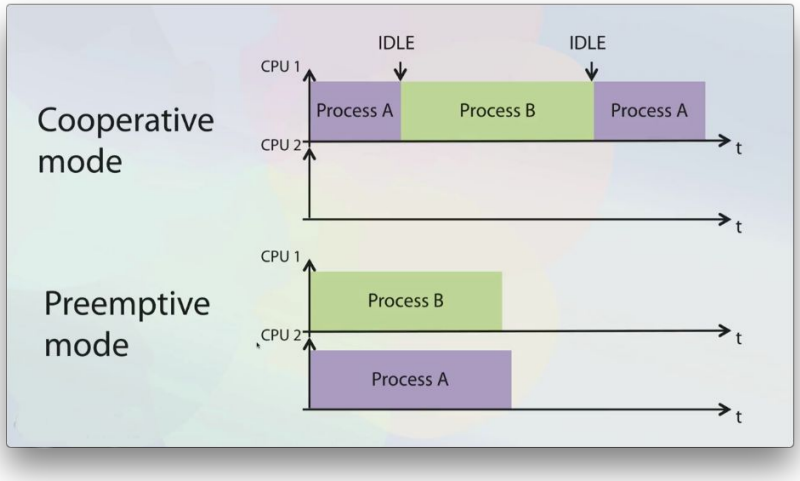
```php
class Foo
{
    final public const XX = "foo";
}

class Bar extends Foo
{
    public const XX = "bar"; // Fatal error
}
```

Fiber

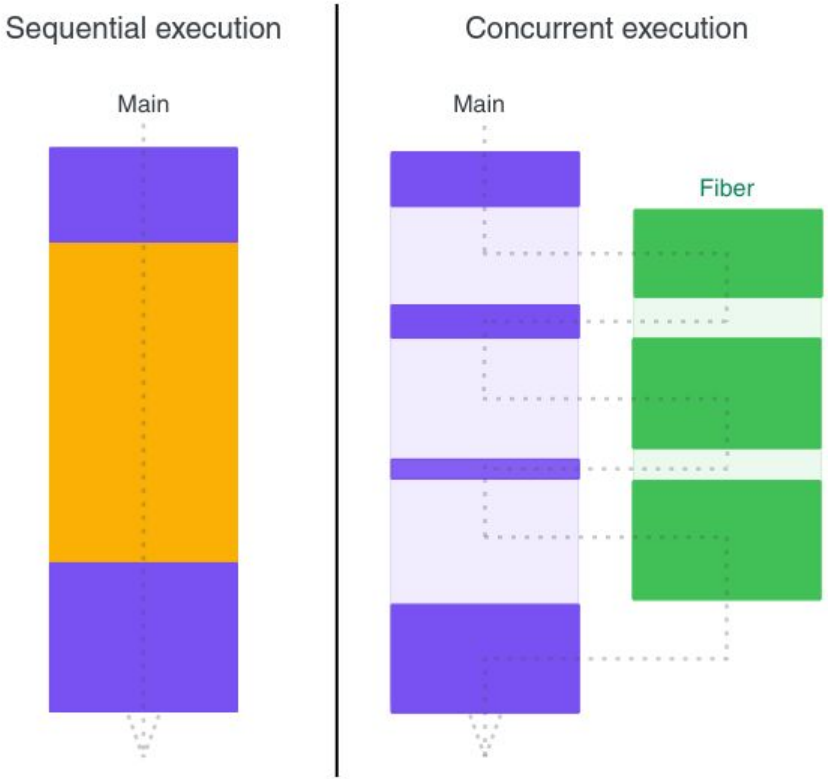
- I **Fiber** implementano il multitasking cooperativo in PHP
- Un Fiber è un blocco di codice che mantiene il proprio stack (variabili e stato)
- Questo codice può essere avviato, sospeso o terminato in maniera cooperativa dal codice principale e dal Fiber

Multitasking cooperativo

- Il **multitasking cooperativo** è una tecnica multitasking che consente a due o più programmi di condividere in modo cooperativo il tempo di elaborazione e le risorse del processore host



Esecuzione concorrente



Non c’è esecuzione in parallelo

- Il codice in un Fiber e nel flusso di esecuzione principale di un’applicazione PHP non vengono eseguiti in parallelo
- E’ compito del flusso di esecuzione principale avviare un Fiber, quando esso entra in azione il suo codice viene eseguito in esclusiva (come se fosse una funzione)
- E’ possibile eseguire un solo Fiber per volta

Esempio

```php
$fiber = new Fiber(function (): void {
    $value = Fiber::suspend('fiber');
    echo "Value used to resume fiber: ", $value, "\n";
});

$value = $fiber->start();
echo "Value from fiber suspending: ", $value, "\n";
$fiber->resume('test');

// Value from fiber suspending: fiber
// Value used to resume fiber: test
```

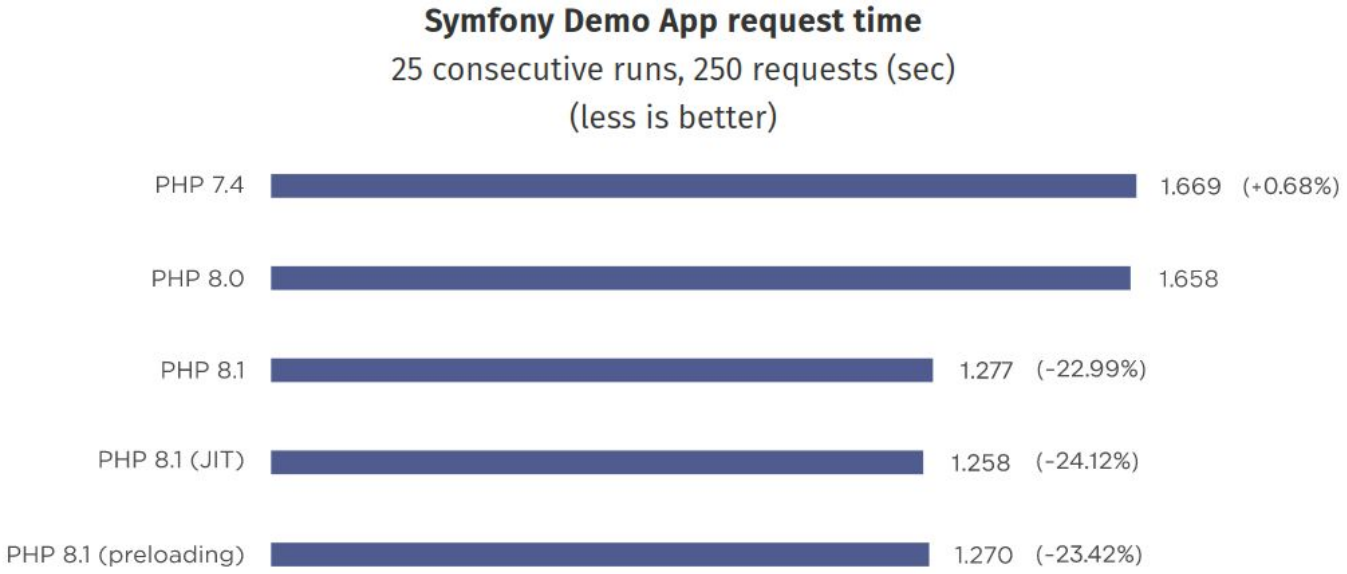
Esempio: copia di file

```php
$files = [
    'source/1.png' => 'dest/a.png',
    'source/2.png' => 'dest/b.png',
    'source/3.png' => 'dest/c.png',
];

$fiber = new Fiber(function(array $files): void {
    foreach($files as $source => $destination) {
        copy($source, $destination);
        Fiber::suspend([$source, $destination]);
    }
});
```

Esempio: copia di file (2)

```
$copied = $fiber->start($files); // [$source, $destination]
$copied_count = 1;
$total_count  = count($files);

while(!$fiber->isTerminated()) {
   $perc = round($copied_count / $total_count, 2) * 100;
   printf("[%d%%]: '%s' to '%s'\n", $perc, $copied[0], $copied[1]);
   $copied = $fiber->resume();
   ++$copied_count;
}
printf("Completed\n");
// [33%]: 'source/1.png' to 'dest/a.png'
// [67%]: 'source/2.png' to 'dest/b.png'
// [100%]: 'source/3.png' to 'dest/c.png'
// Completed
```

Benchmark di PHP 8.1



Riferimenti

- Italo Baeza Cabrera, A Look at the New PHP 8.1 Fibers Feature
- Hatem Ben Yacoub, PHP Fibers, the New Controversial PHP 8.1 Feature Not For You!, PHP Magazine, 2021
- Brent Roose, What's new in PHP 8.1
- Brent Roose, Freek Van der Herten, Front Line PHP, Spatie, e-book
- Doug Bierer, PHP 8 Programming Tips, Tricks and Best Practices, Packt, 2021
- E. Zimuel, Sviluppare in PHP 7, Tecniche Nuove, pp. 432, 2019
- E. Zimuel, The new features of PHP 8.1, PUG Torino, 2021
- E. Zimuel, PHP 8 is coming, Leeds PHP UK, 2020

Grazie!

Contatti: enrico@zimuel.it

Linkedin: https://www.linkedin.com/in/ezimuel/

Twitter: https://twitter.com/ezimuel