

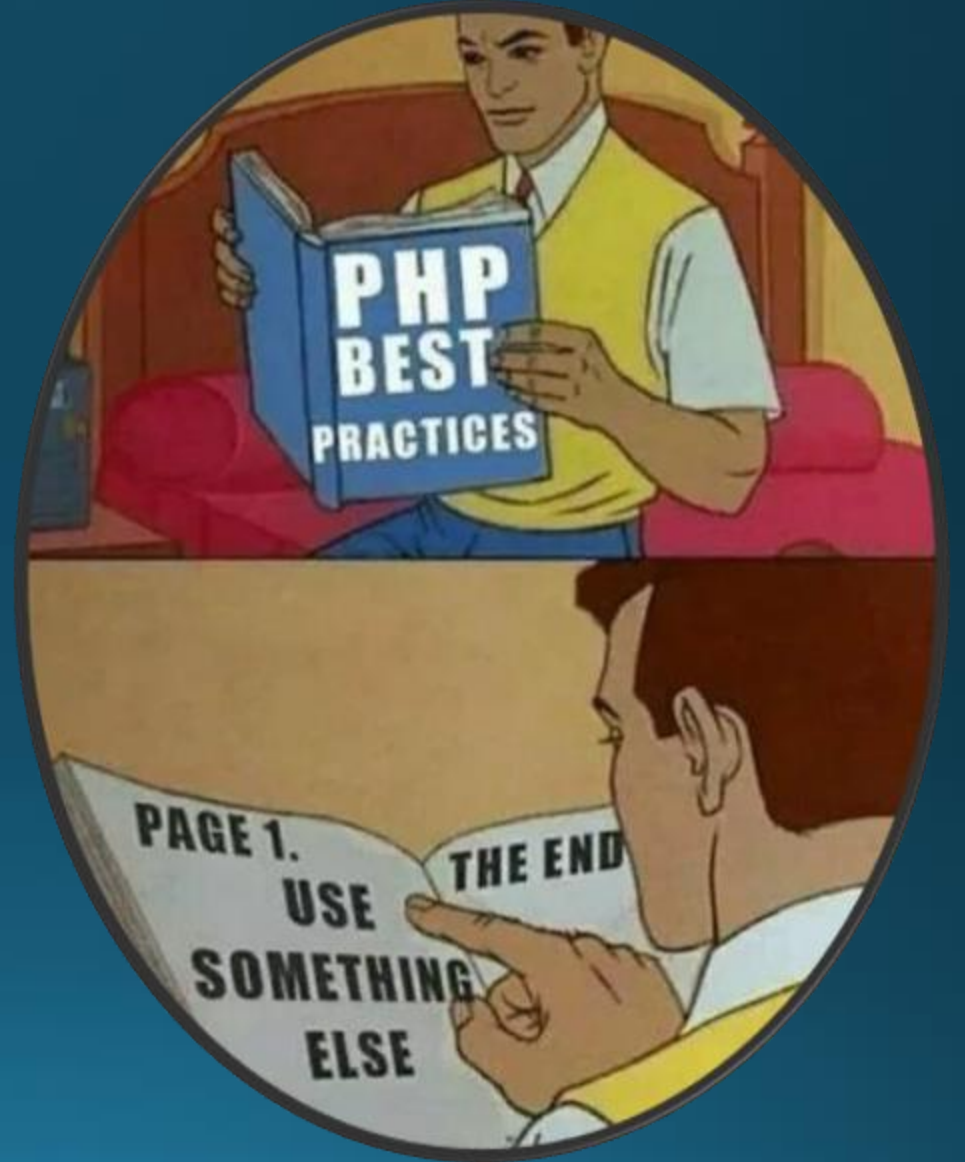
**SWOOLE**

# PHP ile Asenkron Programlama Swoole & PHP 8.4

*Fatih M. Koç*

[github.com/fmkoc](https://github.com/fmkoc)

[linkedin.com/in/fmkoc](https://linkedin.com/in/fmkoc)



# Senkron Programlama

Blocking



---

Görevler ardışık olarak yürütülür.

---

Bir görev tamamlanmadan bir sonraki göreve geçilemez.

---

Örneğin, bir veri tabanı sorgusu veya harici bir API Call yapılırken, uygulama bu işlemin sonucunu beklemek zorundadır ve bu süre zarfında başka hiçbir iş yapamaz.

# Asenkron Programlama

## Non-Blocking



---

Görevler eşzamanlı/bağımsız çalışır.

---

Bir işlem başlatıldığında bu işlemin tamamlanması beklenmeden diğer görevlere devam edebilir.

---

İşlem tamamlandığında, bir «**callback**» veya «**promise**» aracılığıyla sonuç döndürür.

---

Örneğin, asenkron sistemler, sunucuya birden fazla istek gönderebilir ve her birinin yanıtını beklemeden diğer işlere devam edebilir.

---

#1

## Senkron Programlama

## Asenkron Programlama

**Akış**

- Sıralı (Sequential)
- Bir görev bitmeden diğeri başlamaz.

- Eşzamanlı (Concurrent)
- Birden fazla görev aynı anda çalışabilir.

**Bloklama**

- Bloklayan (Blocking)
- I/O operasyonlarında ana iş parçacığı bekler.

- Bloklamayan (Non-Blocking)
- I/O operasyonlarında ana iş parçacığı beklemez.

**Kaynak Kullanımı**

- Daha Az Verimli
- CPU, bir operasyonu beklerken boшта kalabilir.

- Daha Verimli
- CPU, bir operasyonu beklerken başka işler yapabilir.

#2

## Senkron Programlama

## Asenkron Programlama

### Yanıt Süresi

- Daha Yavaş
- Uzun süren görevler uygulamayı yavaşlatır.

- Daha Hızlı
- Genel olarak daha optimizedir.

### Kod Karmaşıklığı

Genellikle Daha Basit

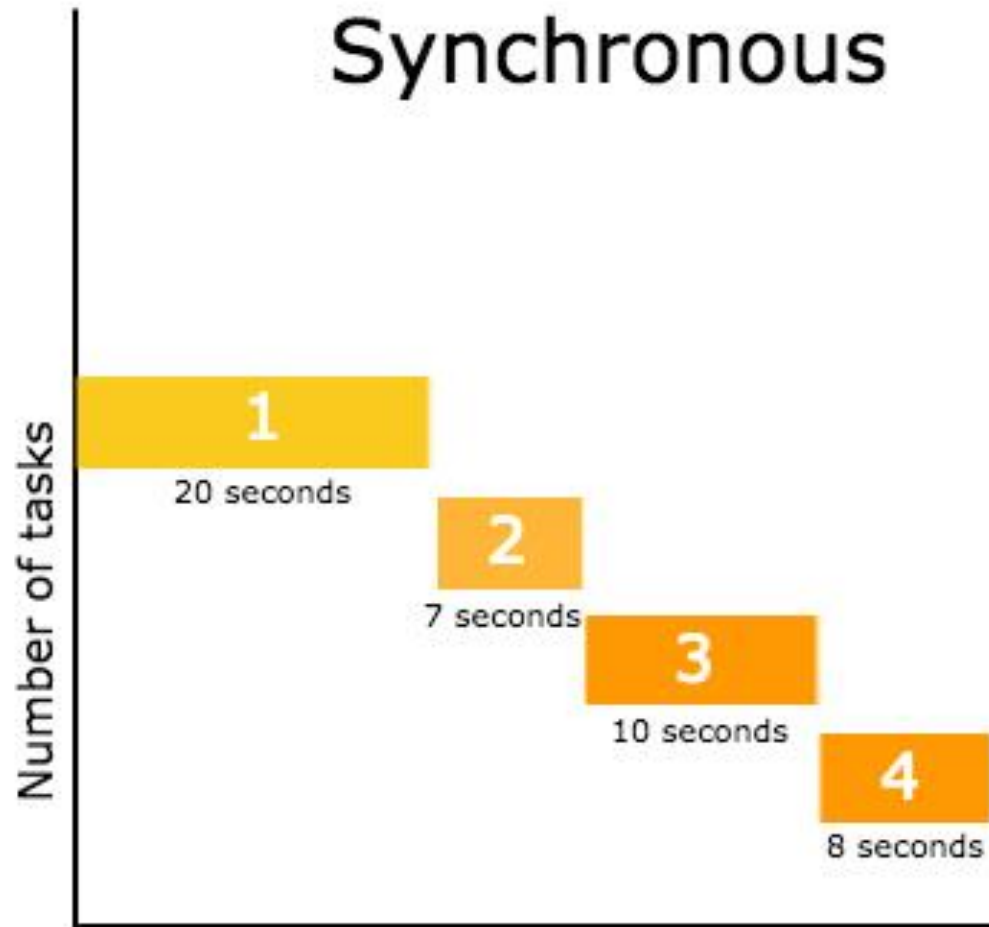
Genellikle Daha Karmaşık  
(Callbacks, promises, coroutines)

### Temel Kullanım Alanları

- CPU Yoğun İşlemler
- Sıradan Web Siteleri

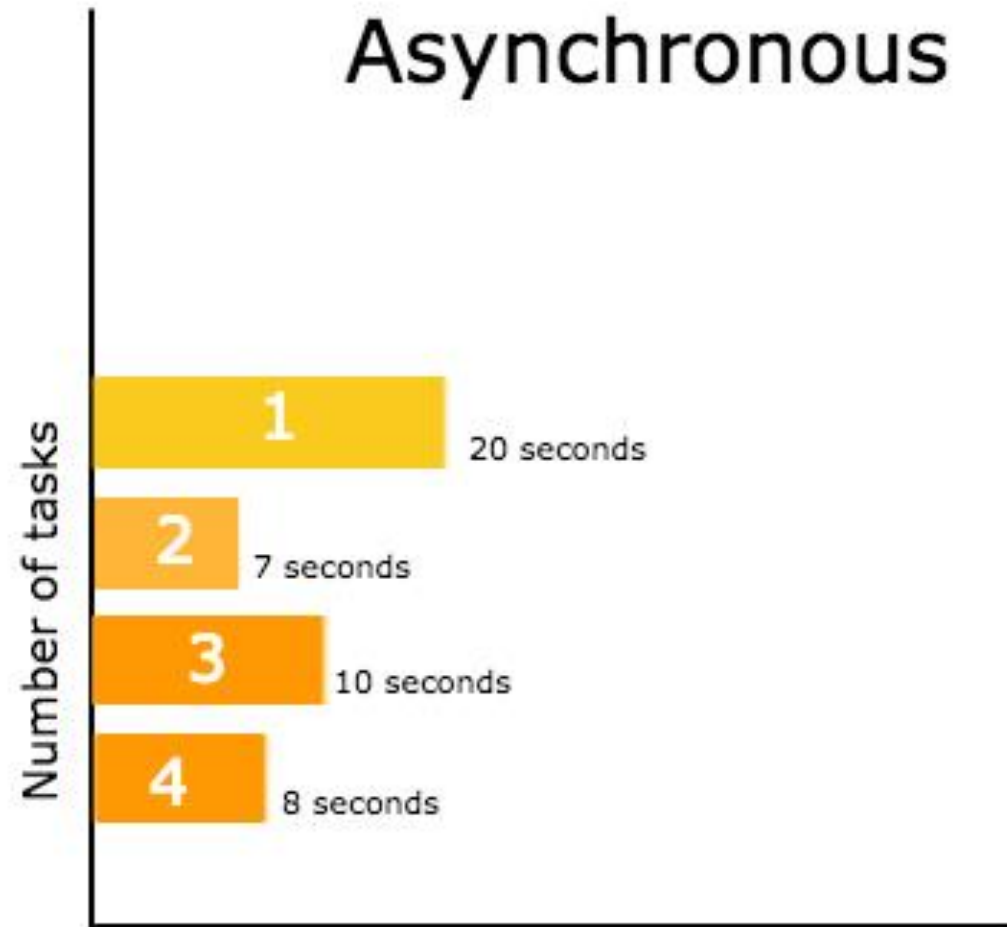
- I/O Yoğun İşlemler
- Gerçek Zamanlı Uygulamalar
- Yüksek Eşzamanlılık

## Synchronous



Total time taken by the tasks.  
**45 seconds**

## Asynchronous



Total time taken by the tasks.  
**20 seconds**



A meme featuring a still from the movie 'Training Day'. Denzel Washington, as the character Alvin Kersh, is shown in a dark suit, pointing a silver handgun directly at the camera with a menacing, shouting expression. The background is a dimly lit room with a doorway visible. The image has rounded corners and is set against a dark blue gradient background.

**TRY TO SAY IT AGAIN**

**"PHP IS NOT TRULY ASYNCHRONOUS"**

# Asenkron Programlama Nerelerde Kullanılır

Ağ İstekleri  
(HTTP/API)

Dosya  
Okuma/Yazma  
(File I/O)

Veri Tabanı  
Sorguları

Olay İşleme  
(Events)

Eşzamanlı  
Görevler  
(Concurrent Jobs)



# Asenkron Mimarilerde Kullanılan Bazı Yaygın Araçlar

RabbitMQ



Redis



AWS SQS




Kayıt sonrası kullanıcıya  
mail gönderme vb.

Distributed sistemlerde  
«session» yönetimi  
API Cache sistemleri vb.

Sipariş oluşturma  
sonrası ürün stoğu  
düşürme, fatura  
oluşturma, ödeme  
işleme vb.

- Bu çözümler programlama yaparken kullanılamıyor
- Cevaplarının takip edilmesi gerekliliği var

# Farklı Dillerde Asenkron Destekleri

- GO -> goroutine 
  - Hafif, thread benzeri işlemler
  - Aynı anda binlercesi çalışabilir

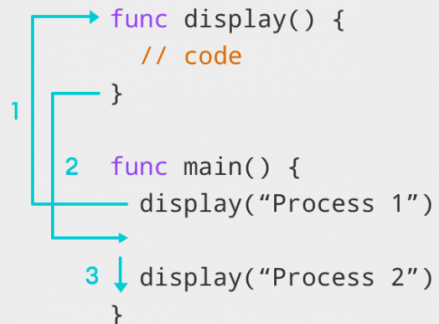
- JS -> async/await

**JS**

- Promise tabanlı, okunabilir yapı
- Tek thread'de yüksek verim sağlar

Normal Program

```
func display() {  
    // code  
}  
  
func main() {  
    display("Process 1")  
    display("Process 2")  
}
```



Program using Goroutine

```
func display() {  
    // code  
}  
  
func main() {  
    go display("Process 1")  
    display("Process 2")  
}
```



```
async function boom() {  
    try {  
        const fireworks = await getFireworks();  
        const trigger = await setUpFireworks(fireworks);  
        return trigger();  
    } catch(error) {  
        return 'Run Away!'  
    }  
}
```

# PHP'de cURL Örneği (Concurrency)

```
$ch1 = curl_init("https://www.gamermarkt.com/tr");  
  
curl_setopt($ch1, CURLOPT_RETURNTRANSFER, true);  
  
$response1 = curl_exec($ch1);  
  
curl_close($ch1);  
  
$ch2 = curl_init("https://kommunity.com/");  
  
curl_setopt($ch2, CURLOPT_RETURNTRANSFER, true);  
  
$response2 = curl_exec($ch2);  
  
curl_close($ch2);
```

```
$mh = curl_multi_init();  
$ch1 = curl_init("https://www.gamermarkt.com/tr");  
$ch2 = curl_init("https://kommunity.com/");  
  
curl_setopt($ch1, CURLOPT_RETURNTRANSFER, true);  
curl_setopt($ch2, CURLOPT_RETURNTRANSFER, true);  
  
curl_multi_add_handle($mh, $ch1);  
curl_multi_add_handle($mh, $ch2);  
  
do {  
    $status = curl_multi_exec($mh, $running);  
} while ($running > 0);  
  
$response1 = curl_multi_getcontent($ch1);  
$response2 = curl_multi_getcontent($ch2);  
  
curl_multi_close($mh);
```

# PHP'deki Yenilikler: Fiber #1

- PHP 8.1 ile yayınlandı.
- PHP 8.3 ile Fibers API daha verimli hale getirildi.
- Thread gibi değil asimetrik coroutine gibi çalışır.
- Bir kod bloğunu çalışırken istenen noktada durdurulabilir (suspend) veya devam ettirilebilir (resume).

A large, stylized graphic of the text 'php 8.1' in a bold, sans-serif font. The 'php' is in a dark blue color, and '8.1' is in a lighter blue color. The text is slightly tilted and has a soft shadow effect.

# PHP'deki Yenilikler: Fiber #2

- Tam bir asenkron deneyimi sunmaz.
- Bazı asenkron işlemleri daha yönetilebilir hale getirir.
- Tam olarak «Bekleme anında diğer işe geçme» davranışı gösterir.

```

function fakeDbQuery(string $query, int $delay): string {
    echo "Query executing: $query\n";
    sleep($delay);
    return "Query executed: $query";
}

$queries = [
    ["query" => "SELECT * FROM users",      "delay"=> 2],
    ["query" => "SELECT * FROM orders",      "delay"=> 3],
    ["query" => "SELECT * FROM products",    "delay"=> 1],
];

$fibers = [];
$results = [];

// Create a Fiber for each query
foreach ($queries as $index => $element) {
    $fibers[$index] = new Fiber(function () use ($element) {
        $result = fakeDbQuery($element["query"], $element["delay"]);
        Fiber::suspend($result); // return the result
    });
}

// Start each Fiber and get results
foreach ($fibers as $index => $fiber) {
    $results[$index] = $fiber->start();
}

// Display results
foreach ($results as $index => $result) {
    echo "[$index] $result\n";
}

echo sprintf("\nTotal execution time: %.2f seconds\n", microtime(true) - $start_time);

```

Query executing: SELECT \* FROM users

Query executing: SELECT \* FROM orders

Query executing: SELECT \* FROM products

[0] Query executed: SELECT \* FROM users

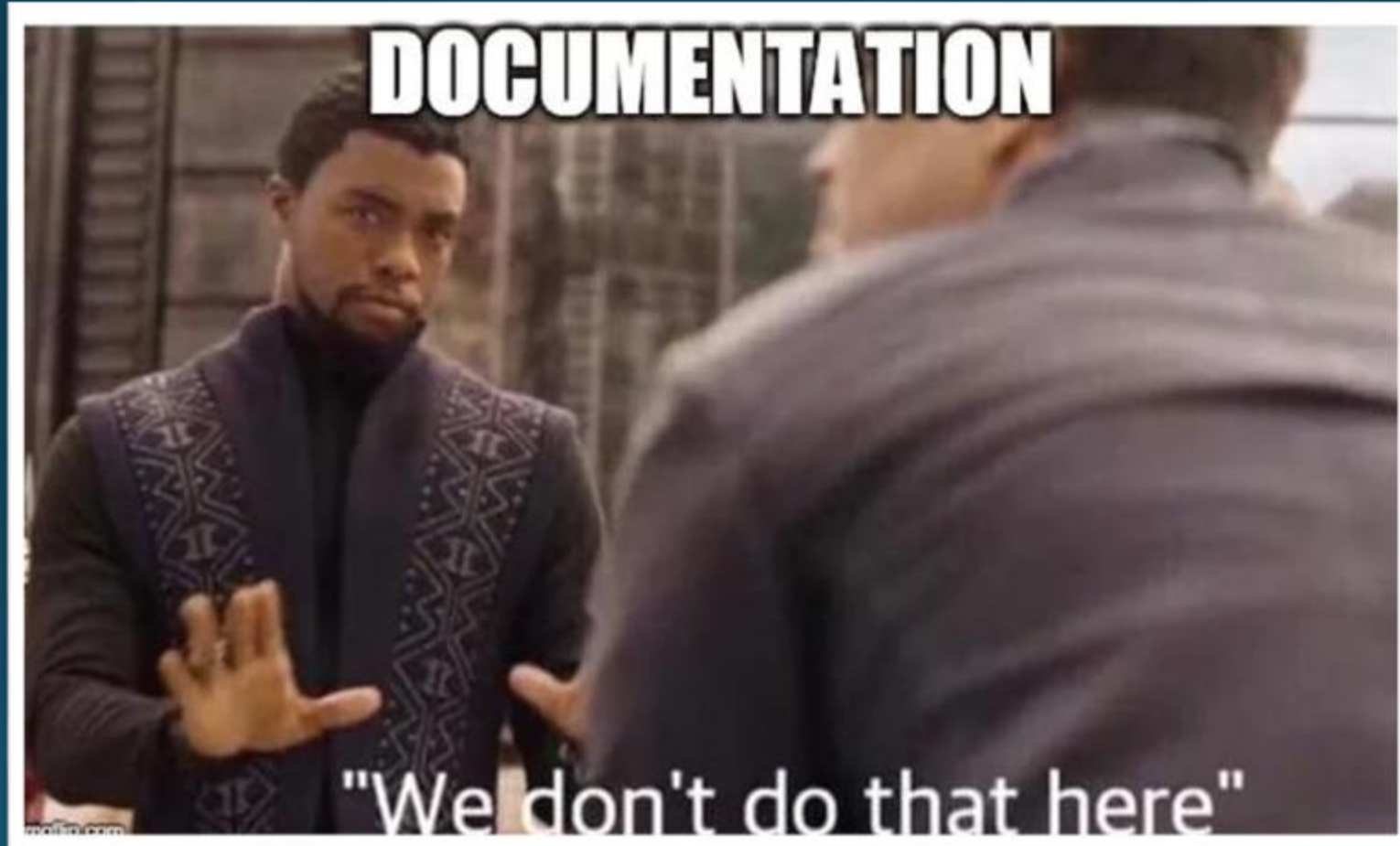
[1] Query executed: SELECT \* FROM orders

[2] Query executed: SELECT \* FROM products

Total execution time: 6.04 seconds



# Kısıtlamalar 🥲



# PHP'de Asenkron Programlama

Swoole





ReactPHP



AMPHP




#1	Swoole	ReactPHP	AMPHP
Geliştirme	C/C++ PHP Extension	PHP PHP Library	PHP PHP Library
Kurulum	PECL 	Composer	Composer
Performans	Yüksek 	Orta*	Orta*
HTTP/WebSocket Server	Var	Yok*	Yok*

#2

Swoole

ReactPHP

AMPHP

Coroutine Desteği	Var (Native)	Yok	Var (Generator Tabanlı)
Event Loop	Var	Var*	Var*
PHP-FPM Bağımlılığı	Yok 	Var	Var
Ayağa Kaldırma	Command Line	Apache/Nginx	Apache/Nginx

# ★ Swoole: Öne Çıkan Özellikler

Yüksek Performans

Gerçek Coroutine Desteği

Server Desteği: HTTP, WebSocket, TCP, gRPC

Microservices & Worker sistemleri için ideal.

# — Swoole: Eksileri



Kurulumu sistem düzeyindedir (PECL)



Shared hosting uyumsuzluğu



Sınırlı framework uyumluluğu (Laravel/Symfony)



Windows için WSL / Docker gerekli.



# Swoole: Detaylar

- 2012 yılında yayınladı.
- C/C++ ile event-driven mimari üzerine geliştirildi.
- 2018 yılında coroutine desteği geldi. (Swoole 4)
- Baş geliştiricisi: Tianfeng Han
- Tencent, Alibaba, Baidu, Xiaomi firmalarında çalışanlar tarafından geliştirildi ve kullanıldı:
  - Gerçek zamanlı uygulamalar
  - Yüksek performanslı API sunucuları
  - Mikroservis mimarileri
  - Oyun sunucuları ve canlı yayın sistemleri
  - Asenkron görev işleme ve zamanlayıcılar

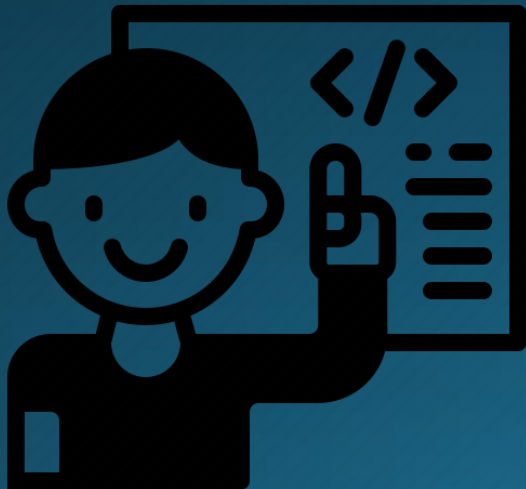


## Swoole: Ne Değildir?



- Geleneksel bir PHP framework değildir.
- Apache/Nginx yerine geçen bir web server değildir.
- Sadece web için değildir. (IoT, TCP/UDP vb.)
- «Magic bullet» değildir.
- Node.js klonu değildir.

## Swoole: Kimler İçin Değildir?



- CRUD tabanlı uygulamalar.
- Yüksek trafikli olmayan web siteleri.
- DevOps bilgisi sınırlı olan kişiler.
- Hızlı prototip geliştirme ihtiyacı olan projeler.

## Swoole: Neler Yapılabilir?



- Gerçek zamanlı uygulamalar (Sohbet, anlık bildirim, işbirliği araçları)
- Yüksek performanslı API'lar ve API Gateway
- WebSocket uygulamaları (Oyun sunucusu, gerçek zamanlı panolar)
- Mikroservis uygulamaları (Service mesh, dağıtık sistemler)
- IoT uygulamaları (Hub, sensör veri işleme)

吾有一術。名之曰「快排」。欲行是術。必先得一列。曰「甲」。乃行是術曰。

若「甲」之長弗大於一者。

乃得「甲」

也。

吾有三列。名之曰「首」。曰「領」。曰「尾」。

夫「甲」之一。名之曰「甲一」。

充「領」以「甲一」。

夫「甲」之其餘。名之曰「甲餘」。

凡「甲餘」中之「丁」。

若「丁」小於「甲一」者。

充「首」以「丁」。

若非。

充「尾」以「丁」

也。

云云。

施「快排」於「首」。昔之「首」者。今其是矣。

施「快排」於「尾」。昔之「尾」者。今其是矣。

銜「首」以「領」以「尾」。名之曰「乙」。

乃得「乙」。

是謂「快排」之術也。

吾有一列。名之曰「己」。

充「己」以五。以三。以二十。以八。以三十五。以七百。

施「快排」於「己」。書之。

# Swoole vs OpenSwoole



- Bir fork olarak ortaya çıkmıştır.
- İngilizce belgelerin yetersizliği sebep olmuştur.
- Aktif olarak geliştirilmektedir.
- OpenSwoole'un «Docker» desteği ve «Composer» paketleriyle uyumu nispeten daha iyidir.
- Hyperf Framework Swoole kullanır.
- Laravel Octane ve Symfony Swoole Bundle OpenSwoole kullanır.

# Tanım: Event Loop

---

Linux'taki `epoll_wait` fonksiyonuna dayanan düşük seviyeli bir API'dir

---

Bu yapı, dosya tanımlayıcıları (file descriptor - fd) üzerinden I/O (giriş/çıkış) olaylarını izler.

---

Örneğin, bir soket üzerinden veri geldiğinde veya bir dosya yazılabilir hale geldiğinde, event loop bu olayları algılar ve callback fonksiyonlarını tetikler .

```
// Connect to a TCP server
$fp = stream_socket_client("tcp://example.com:80", $errno, $errstr, 30);

// Send HTTP GET Request
fwrite($fp, "GET / HTTP/1.1\r\nHost: example.com\r\nConnection: close\r\n\r\n");

// Register the file pointer with OpenSwoole's event loop
OpenSwoole\Event::add($fp, function($fp) {
    while (!feof($fp)) {
        $resp = fread($fp, 8192);
        if ($resp === false) {
            break;
        }
        echo $resp;
    }

    // Remove the file pointer from the event loop and close it
    OpenSwoole\Event::del($fp);
    fclose($fp);
});
```



# Tanım: Coroutine

---

İşletim sistemi düzeyinde değil, kullanıcı düzeyinde çalışan hafif iş parçacıklarıdır.

---

Düşük bellek kullanımıyla yüksek eşzamanlılık sağlarlar.

---

Bloklayıcı I/O işlemlerini asenkron hale getirerek, uygulamanın diğer işlemleri engellemeden çalışmasını sağlarlar.

---

Swoole'da her coroutine, yaklaşık 8KB bellek kullanır, bu da binlerce coroutine'in aynı anda çalışmasını mümkün kılar. (Kaynak: OpenSwoole Docs)

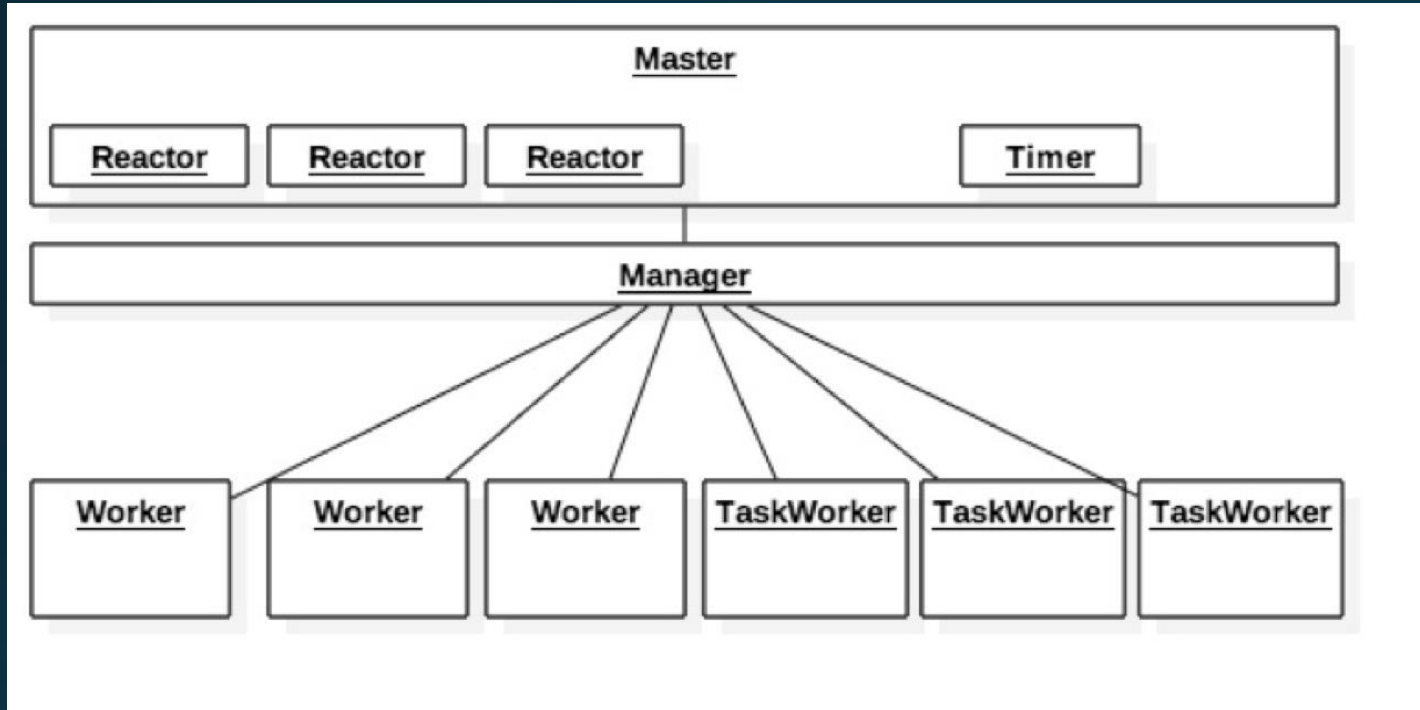
```
use OpenSwoole\Coroutine as Co;

co::run(function()
{
    go(function()
    {
        // Runs immediately
        var_dump(file_get_contents("http://openswoole.com/"));
    });

    go(function()
    {
        // Waits once second before continuing
        Co::sleep(1);
        var_dump(file_get_contents("http://www.google.com/"));
    });
});

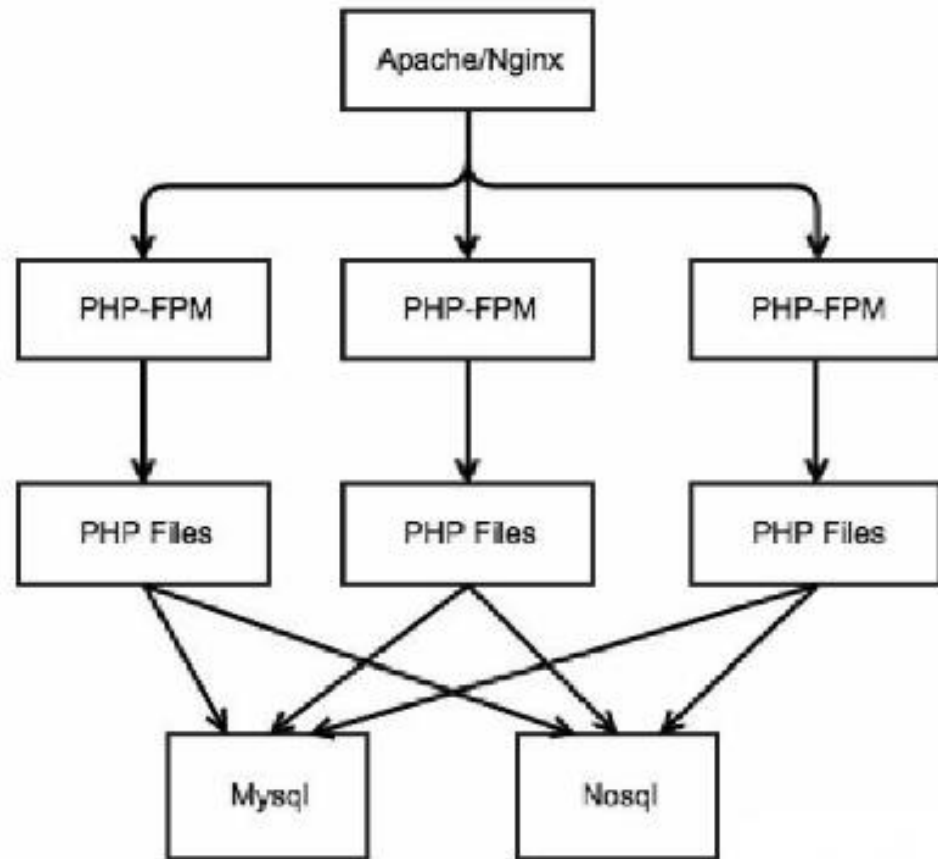
echo "Outside any Coroutine Context.\n";
```

# Swoole Çalışma Mantığı

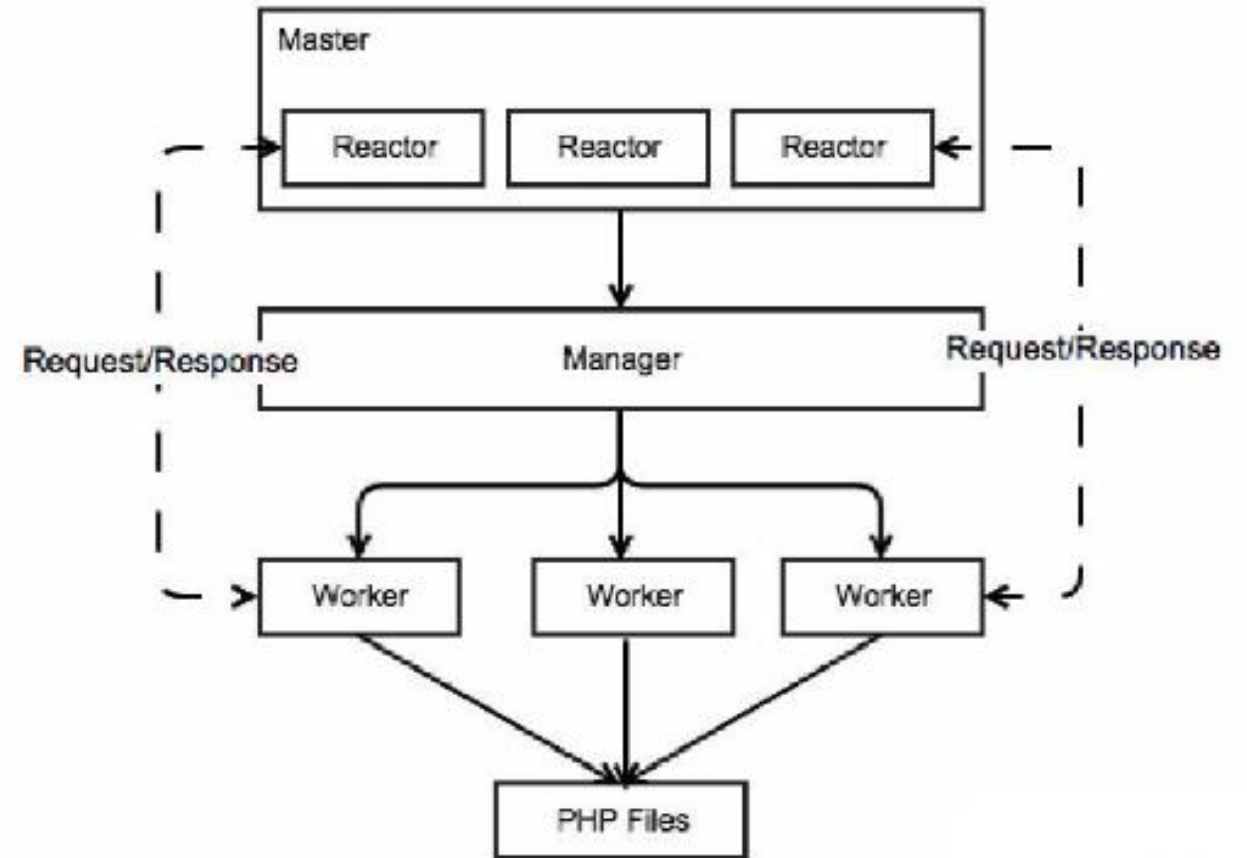


- Event-driven & multi-process
- Swoole'da başlıca dört ana süreç (process) rolü vardır:
  - **Master Process,**
  - **Reactor**
  - **Manager Process**
  - **Worker**
    - **Task Worker**

## Apache / Nginx + PHP-FPM



## Swoole



# Master Process (1/4)

- Swoole uygulamasını başlatan ana süreçtir.
- Görevleri:
  - Konfigürasyon ve başlatma işlemleri.
  - Listener'ları ayarlar ve portları açar.
  - Alt süreçlerin (worker, manager, reactor) yönetimini ve yeniden başlatılmasını kontrol eder.
  - Ağdan gelen bağlantıları kabul eder ve bunları uygun **reactor** süreçlerine yönlendirir.
- PHP kodu **master process** içinde çalışmaz. Sadece kontrol ve yönetim görevi vardır.

# Reactor Process (2/4)

- Her bir reactor, TCP/UDP bağlantılarını ve ağ olaylarını asenkron olarak işler.
- Görevleri:
  - Ağdan gelen veri okuma/yazma, bağlantı yönetimi ve olayları algılar.
  - Bağlantılardan gelen istekleri ilgili worker süreçlerine iletir.
  - Genellikle çekirdek (core) sayısı kadar reactor süreci başlatılır.
- Özetle; event loopun asıl döndüğü yerdir; çok hızlıdır ve CPU'da asılı kalmaz.

# Manager Process (3/4)

- Worker ve task worker süreçlerinin başlatılmasından ve izlenmesinden sorumludur.
- **Görevi:** Bir worker veya task worker kapandığında yerine yenisini başlatır.
- Bu süreçte de PHP kodu çalışmaz, sadece süreç kontrolü yapar.



# Worker Process (4/4)

- Asıl PHP uygulama kodunun çalıştığı süreçlerdir.
- Görevleri:
  - Ağdan gelen istekleri işler. Kullanıcıdan gelen veriyi işler, yanıt oluşturur ve sonucu istemciye gönderir.
  - Çok sayıda worker süreci aynı anda çalışabilir, bu sayede paralel istekler işlenir.
  - Her worker süreci birbirinden bağımsızdır (bellek paylaşmaz).

Task Worker: Worker bir iş uzun sürecekseniz üzerindeki görevi Task worker'a «devreder», böylece ana iş akışını bloke etmeden devam eder.



# Değişkenlerin Scope Problemi & Process Kapanınca Yok Olması

- Her worker değişkeni kendi belleğinde tutar, paylaşım yoktur.
- Bir worker kapanınca tanımlı değişkenler de yok olur.

```
$server = new OpenSwoole\Server("127.0.0.1", 9501);

$counter = 0;

$server->on("receive", function (OpenSwoole\Server $server, int $fd, int $reactor_id, string $data) use (&$counter) {
    $counter++;
    $server->send($fd, "Counter for this worker: $counter\n");
});

$server->start();
```



# Blocking Fonksiyonlardan Kaçınma Gereksinimi

```
//Blocking
OpenSwoole\Server::on('receive', function($serv, $fd, $reactor_id, $data) {
    $result = file_get_contents('http://slow-api.com/data');
    $serv->send($fd, "Result: $result\n");
});

//Non-blocking
OpenSwoole\Server::on('receive', function($serv, $fd, $reactor_id, $data) {
    go(function() use ($serv, $fd) {
        $client = new OpenSwoole\Coroutine\Http\Client('slow-api.com', 80);
        $client->set(['timeout' => 5]);
        $client->get('/data');
        $result = $client->getBody();
        $client->close();
        $serv->send($fd, "Result: $result\n");
    });
});
```

# Swoole Runtime Hook #1

- PHP'nin yerleşik olan blocking fonksiyonlarını non-blocking hale getirir ve coroutine olarak çalışmasını sağlar.
- Fonksiyonları alt seviyelerde yakalar ve onları manipüle eder.
- Mevcut kodda hiçbir değişiklik yapmadan uygulanabilir.
- Bazı örnekler:
  - sleep()
  - file\_get\_contents()
  - fopen()
  - stream\_socket\_client
  - PDO library
  - cURL functions





# Swoole Runtime Hook #2

- Aşağıdaki örnekte «hook» tüm fonksiyonlar için aktifleştirilmiştir.
- Bazı Flag'ler:
  - HOOK\_CURL
  - HOOK\_SLEEP
  - HOOK\_FILE

```
OpenSwoole\Runtime::enableCoroutine(true, OpenSwoole\Runtime::HOOK_ALL);

Co\run(function () {
    $data = file_get_contents("https://phpkonf.org");
    echo $data;
});
```

# OpenSwoole\Table

Tüm Worker'lar Arası Paylaşımlı Bellek

```
$table = new OpenSwoole\Table(1024);  
$table->column('counter', OpenSwoole\Table::TYPE_INT);  
$table->create();  
  
$server->on('receive', function ($server, $fd, $reactor_id, $data) use ($table) {  
    $table->incr('my_key', 'counter');  
    $count = $table->get('my_key', 'counter');  
    $server->send($fd, "Global counter across all workers: $count\n");  
});
```

# OpenSwoole\Atomic

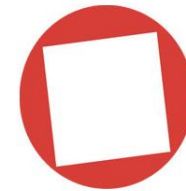
## Tüm Worker'lar Arası Ortak Sayaç

```
$atomic = new OpenSwoole\Atomic(0);

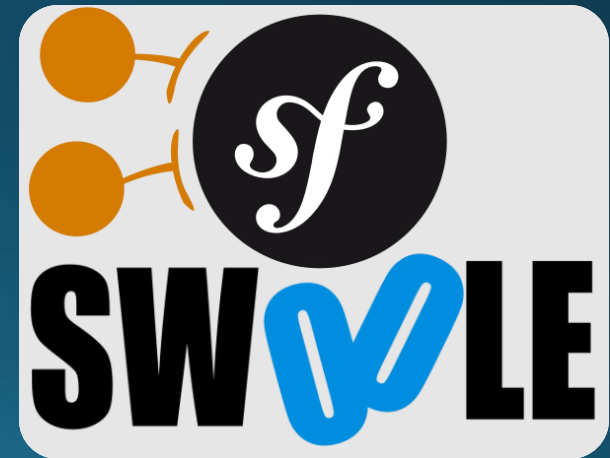
$server->on('receive', function ($server, $fd, $reactor_id, $data) use ($atomic) {
    $count = $atomic->add(1);
    $server->send($fd, "Atomic counter: $count\n");
});
```

# Frameworkler ile Kullanımı

- Laravel: Laravel Octane (Official)
  - <https://laravel.com/docs/12.x/octane>
- Symfony: Symfony Swoole Bundle
  - <https://github.com/symfony-swoole/swoole-bundle>



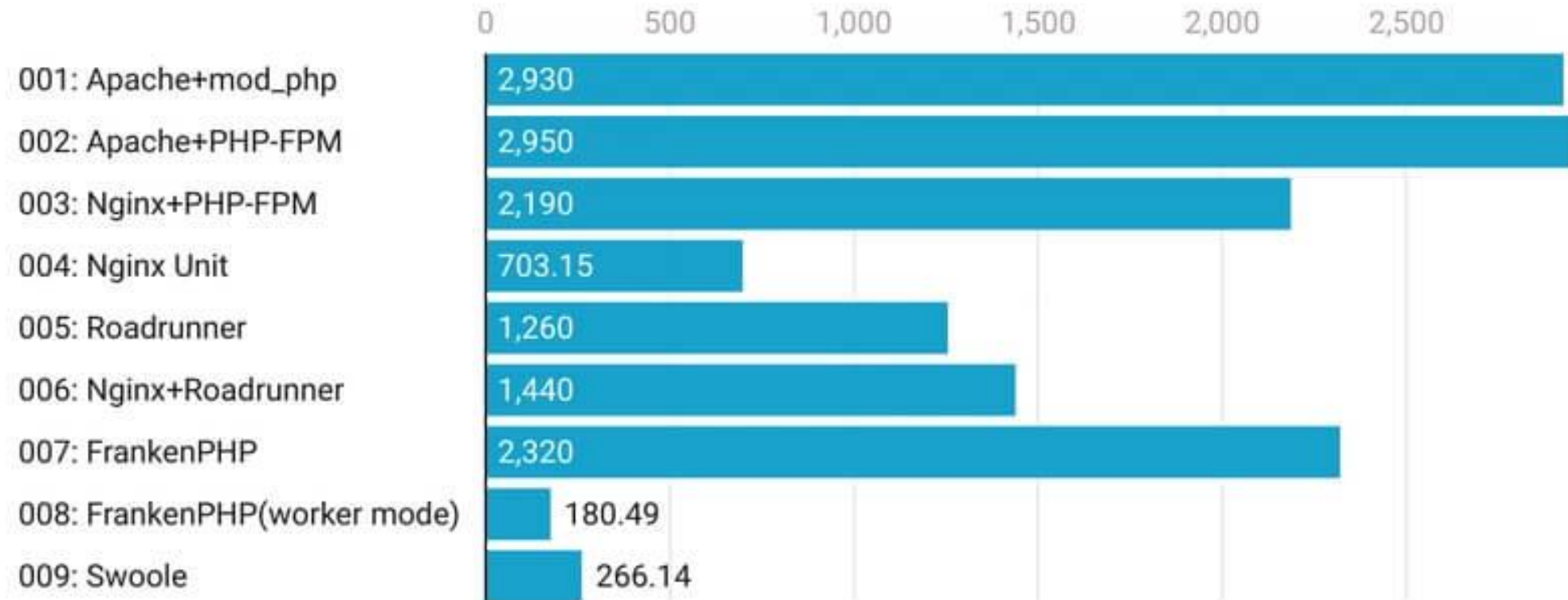
Laravel Octane





# Benchmark #1 (2024)

## Average response time(ms), concurrency 1000



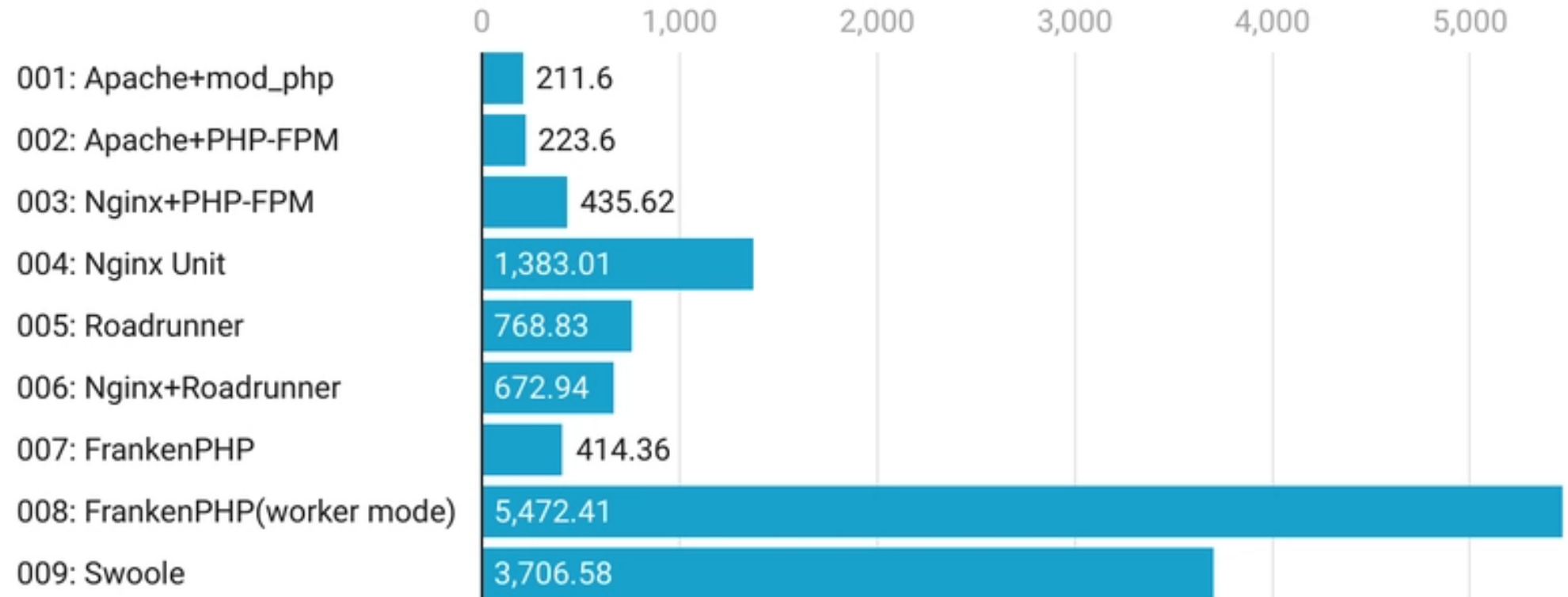
Docker container: CPU: 1, RAM: 1GB. Testing utility: K6(<https://k6.io/open-source/>)

Source: GitHub • Created with Datawrapper

<https://dev.to/dimdev/performance-benchmark-of-php-runtimes-2lmc>

# Benchmark #2 (2024)

## Average requests per second, concurrency 1000



Docker container: CPU: 1, RAM: 1GB. Testing utility: K6(<https://k6.io/open-source/>)

Source: GitHub • Created with Datawrapper

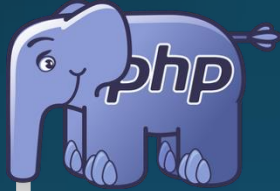
<https://dev.to/dimdev/performance-benchmark-of-php-runtimes-2lmc>

# OpenSwoole 25 & PHP 8.4 ile Demo Proje

- Simple HTTP Server
- XOX (Tic Tac Toe) Game Server

 [github.com/fmkoc/phpkonf-2025](https://github.com/fmkoc/phpkonf-2025)

# Dinlediğiniz için Teşekkürler



*Fatih M. Koç*

[github.com/fmkoc](https://github.com/fmkoc)

[linkedin.com/in/fmkoc](https://linkedin.com/in/fmkoc)

