PHP-Code profilieren

PHP-Code in Exponential messen und beschleunigen - exp:benchmark für Seiten, Kernel-Messpunkte und Mikro-Benchmarks, eZDebug-Akkumulatoren um eigenen Code, die SQL- und Speicherwerte der Debug-Leiste, das…

Source file: `doc/ger-de/hilfe/php-entwickler/php-code-profilieren.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/profiling-php-code.md

Diese Seite richtet sich an PHP-Entwickler, die Code schneller machen oder nachweisen müssen, dass eine Änderung ihn nicht langsamer gemacht hat. Die Methode ist immer dieselbe: messen, eine Sache ändern, erneut messen, unter denselben Bedingungen. Die Anwendungssicht ist Profiling und Performance; der vollständige Leitfaden ist Exponential benchmarken.

Inhalt

1. Drei Messebenen
2. Seiten: exp:benchmark
3. Der Kernel und Mikro-Benchmarks
4. Innerhalb einer Anfrage
5. OPcache, unter jeder Engine
6. Der SQL-Abfrage-Cache
7. Wiederkehrende Ursachen
8. Fair messen

Drei Messebenen

Ebene	Werkzeug	Beantwortet
ganze Seiten über HTTP	`exp:benchmark`	ist die Site schneller oder langsamer, und um wie viel
Kernel-Operationen im Prozess	`exp:benchmark kernel`	welcher Teil einer Anfrage kostet (Boot, Rendern, INI, Fetch, Datenbank, Cache)
Ihr Code in einer Anfrage	Akkumulatoren, die Debug-Leiste	wo in Ihrem Code die Zeit verbraucht wird

Seiten: exp:benchmark

Shell

```
php bin/php/console exp:benchmark --save=var/benchmark/base.json
# change something
php bin/php/console exp:benchmark --baseline=var/benchmark/base.json --threshold=15
php bin/php/console exp:benchmark --cold --requests=50                # past the response caches
php bin/php/console exp:benchmark --compare=https://www.example.com,https://www.example.com:8080
```

Es fragt eine Reihe von Seiten nach einem Warm-up viele Male ab und gibt Median, p90, p99

und mehr aus. `--baseline` mit `--threshold` macht daraus eine Regressionsprüfung, die fehlschlägt, wenn eine Seite um mehr als den Schwellenwert langsamer wurde (Benchmark).

Der Kernel und Mikro-Benchmarks

Shell

```
php bin/php/console exp:benchmark kernel --repeat=30 --save=var/benchmark/kernel.json
php bin/php/benchmark.php micro --baseline=tests/benchmark/baseline.json --all-checks
```

Die Kernel-Messpunkte messen im Prozess Boot, Rendern (kalt und warm), Laden der INI, einen Inhalts-Fetch, einen Datenbank-Roundtrip sowie Schreib- und Lesezugriffe auf den Cache. Der Mikro-Benchmark misst reine PHP-Hot-Paths ohne HTTP oder Datenbank; der Performance-Workflow der CI führt ihn unter PHP 8.5 mit ausgeschaltetem OPcache gegen `tests/benchmark/baseline.json` aus, als Warnung.

Innerhalb einer Anfrage

Umschließen Sie den verdächtigen Code mit einem Akkumulator:

PHP

```
eZDebug::accumulatorStart( 'acmereport_totals', 'acmereport_total', 'Monthly totals' );
$totals = acmeReportService::monthlyTotals( $month );
eZDebug::accumulatorStop( 'acmereport_totals' );
```

Mit eingeschalteter Debug-Ausgabe zeigt die Leiste:

- **Zeiten**: Ihre Akkumulatoren mit Gesamtzeit und Anzahl der Aufrufe, neben denen des Kernels;
- **SQL**: die Anzahl der Anweisungen, ihre Zeit und ihren Anteil an der Seite, die langsamsten zuerst (`SQLOutput=enabled`);
- **Speicher**: die Spitze und die Schritte, die am meisten verbraucht haben;
- **Templates**: wie viele Templates verwendet wurden.

Die Zusammenfassung färbt Werte anhand der Schwellenwerte in `debugbar.ini` gelb oder rot, wodurch ein neuer Hot Spot auf einen Blick sichtbar wird (Die Debug-Leiste für PHP-Entwickler).

OPcache, unter jeder Engine

Der Kernel kann pro Anfrage eine Zeile schreiben, die angibt, wie die Anfrage OPcache genutzt hat: Treffer und Fehlschläge, die am häufigsten ausgelieferten Skripte und eingebundene Dateien, die OPcache nicht hält (jede davon wird bei jedem Einbinden erneut kompiliert). Das ist ausgeschaltet, sofern die Datei `var/tmp/opcache_profile.on` nicht existiert; die Zeilen gehen nach `var/tmp/opcache_profile.log` (`ezpKernelWeb::writeOPcacheProfile()`). Nutzen Sie es, um Engines zu vergleichen; entfernen Sie die Schalterdatei danach (Velocity und der Opcode-Cache).

Der SQL-Abfrage-Cache

`settings/querycache.ini` (standardmäßig `Mode=off`) speichert die Ergebnisse wiederholter Leseabfragen zwischen (standardmäßig in APCu, wo der Prozess es hat, sonst in Dateien).

Messen Sie mit ein- und ausgeschaltetem Cache: Eine Seite, die dieselbe Abfrage viele Male stellt, gewinnt, eine Seite mit eindeutigen Abfragen nicht. Er gilt nicht für MongoDB (SQL-Abfrage-Cache).

Wiederkehrende Ursachen

Symptom	Übliche Ursache	Lösung
viele Abfragen pro Seite	Fetches in Schleifen, `dataMap()` bei jedem Element einer Liste	einmal abrufen; begrenzen; ein Cache-Block
eine Liste wird mit den Daten langsamer	kein Limit oder eine Zählung von allem	seitenweise; begrenzte Zählungen
der Speicher wächst in einem Befehl	Objekte bleiben in den In-Memory-Caches des Kernels	`eZContentObject::clearCache()` zwischen den Seiten
ein langsames Veröffentlichen	schwere Arbeit in einem Workflow-Ereignis oder Listener	eine Warteschlange oder ein Content-Job
langsam nach dem Deployment	Caches geleert	wärmen Sie sie vor (OPcache und Cache-Vorladen)
nach einer Änderung nur unter Velocity langsam	Worker nicht neu gestartet oder OPcache validiert nicht neu	`exp:velocity deploy`

Fair messen

- Dieselbe Maschine, nichts anderes läuft, OPcache wie in der Produktion, Xdebug aus.
- Vergleichen Sie zwei Server mit abwechselnden Anfragen, nicht indem Sie einen nach dem anderen messen.
- Halten Sie bei jedem Lauf die Umgebung fest; ein Vergleich mit einer Baseline aus einem anderen Aufbau sagt wenig.
- Bewahren Sie die Läufe auf: Committen Sie eine Baseline-Datei für Ihre Erweiterung und vergleichen Sie vor einem Release damit.

Verwandte Seiten

- Exponential benchmarken
- Performance in Erweiterungen
- Caching in Anwendungen
- Freie Dokumentation: diese Seite darf unter der GNU FDL 1.3 frei kopiert, übersetzt und verbessert werden.

> **Werden Sie Exponential-Partner**
>
> Partner teilen ihre Informationen, stellen Mittel bereit, geben Beiträge zurück und entwickeln die Produkte mit uns, und so wächst Exponential. Eine Partner-Website und ein Dashboard sind für Mitte 2027 geplant: Werden Sie noch heute Partner, damit wir schneller wachsen und neue Funktionen früher liefern.
>
> Partner: hier beginnen | Wie Sie sich bewerben

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.