Persistente Velocity-Worker für PHP-Entwickler

Was sich für PHP-Code ändert, wenn Exponential unter Velocity läuft, dem empfohlenen Anwendungsserver - Worker, die einmal booten und viele Anfragen bedienen, das Warm-up, was der Kernel zwischen Anfragen zurücksetzt...

Source file: `doc/ger-de/hilfe/php-entwickler/persistente-velocity-worker-fuer-php-entwickler.md`
Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/velocity-persistent-workers-for-php-developers.md

Diese Seite richtet sich an PHP-Entwickler, deren Code unter Exponential Velocity läuft, dem Anwendungsserver, der ab 6.0.15 Teil von Exponential und die empfohlene Engine für jede Stufe ist (Velocity ist Teil von Exponential 6.0.15). Exponential wurde für einen PHP-Prozess pro Anfrage gebaut; Velocity bootet einmal und hält einen Pool von Workern, die jeweils viele Anfragen bedienen. Der meiste Code läuft unverändert. Diese Seite listet auf, was nicht, und warum.

Inhalt

1. Das Modell
2. Was Kernel und Engine für Sie zurücksetzen
3. Was Sie nicht tun dürfen
4. Arbeit pro Anfrage: eZExecution::requestKey()
5. cleanExit() beendet die Anfrage mit einer Exception
6. Neue Klassen und neue Einstellungen
7. Eine PHP-Änderung bereitstellen
8. Ihren Code für Velocity testen

Das Modell

	PHP-FPM, mod_php	**Velocity (Engine `qbix`)**
Prozess pro Anfrage	frisch	ein Worker, der zwischen Anfragen erhalten bleibt (`[ServerSettings] ForkPerRequest=disabled`, der Standard seit dem 30. September 2026)
Boot	bei jeder Anfrage	einmal, im Warm-up, dann von den Workern geerbt
statische Eigenschaften	nach der Anfrage weg	**erhalten**, sofern nicht zurückgesetzt
geladene Klassen	pro Anfrage geladen	für die Lebensdauer des Workers erhalten
Einstellungen	pro Anfrage gelesen (zwischengespeichert)	einmal pro Worker gelesen

	PHP-FPM, mod_php	Velocity (Engine `qbix`)
SAPI	`fpm-fcgi`, `apache2handler`	die Kommandozeilen-SAPI; die Engine sorgt dafür, dass `header()`, `setcookie()` und verwandte Funktionen arbeiten

Es gibt zwei weitere Engines: FrankenPHP im klassischen Modus (ein sauberer Zustand pro Anfrage, wie PHP-FPM) und den eingebauten Server von PHP für die Entwicklung (Velocity-Engines, FrankenPHP für PHP-Entwickler).

Was Kernel und Engine für Sie zurücksetzen

- **Statische Eigenschaften von Klassen.** Zwischen Anfragen setzt ein Worker die statischen Eigenschaften jeder Klasse auf den Stand nach seinem Warm-up zurück. Die Engine liest die deklarierten Standardwerte aus dem Quellcode.
- **Shutdown-Funktionen und Handler.** Die Shutdown-Funktionen, die eine Anfrage registriert hat, werden ausgeführt und dann vergessen; die Aufräum- und Fatal-Error-Handler von `eZExecution`, sein Flag für ein sauberes Ende und seine eigenen Shutdown-Funktionen gehören zu einer Anfrage.
- **Superglobals und die Anfrage.** `$_GET`, `$_POST`, `$_COOKIE`, `$_SERVER` und `$_SESSION` gelten pro Anfrage; `REQUEST_TIME` und `REQUEST_TIME_FLOAT` gelten pro Anfrage (Engine 0.0.4.42 und neuer).
- **Die Reste des Warm-ups.** Ab Engine 0.0.4.51 wird die Anfrage des Warm-ups beendet, bevor die Worker geforkt werden: Ihre Header, Handler, Sitzung, ini-Änderungen und Ausgabepuffer werden zurückgesetzt.

Was Sie nicht tun dürfen

Muster	Warum es unter Workern bricht	Stattdessen
`static $cache` auf Funktionsebene, nach nichts geschlüsselt	überlebt zwischen Anfragen: Die Daten eines Siteaccess erreichen die nächste Anfrage	eine statische Klasseneigenschaft (vom Worker zurückgesetzt), geschlüsselt nach dem, wovon sie abhängt
ein `true`-Flag, um „einmal zu registrieren“	das Flag bleibt aus dem Warm-up erhalten; die zweite Anfrage registriert nichts	schlüsseln Sie es nach `eZExecution::requestKey()`
einer statischen Klasseneigenschaft nach dem Klassenrumpf einen Wert zuweisen (`acmeFoo::$list = ...;` auf Dateiebene)	der Worker setzt statische Eigenschaften auf den deklarierten Wert zurück, und `include_once` führt die Zuweisung nicht erneut aus	deklarieren Sie den Wert im Klassenrumpf
`$_POST`, `$_GET` lesen, bevor der Kernel sie vorbereitet hat	veraltet oder leer	`eZHTTPTool`
ein langes `while ( ob_get_level() > 0 ) ob_end_clean();`	der Worker hält einen Puffer, der sich nicht entfernen lässt; die Schleife endet nie	`eZExecution::discardOutputBuffers()`

Die Einzelheiten und Beispiele finden Sie unter Fallen durch statischen Zustand.

Arbeit pro Anfrage: eZExecution::requestKey()

Sichern Sie eine Registrierung, die einmal pro Anfrage geschehen muss, mit dem Schlüssel der Anfrage ab (ihrem `REQUEST_TIME_FLOAT`):

PHP

```php
if ( self::$registered !== eZExecution::requestKey() )
{
    self::$registered = eZExecution::requestKey();
    register_shutdown_function( array( __CLASS__, 'flush' ) );
}
```

`eZExecution::addCleanupHandler()` und `addFatalErrorHandler()` dürfen einfach bei jeder Anfrage aufgerufen werden: Derselbe Handler wird nur einmal behalten. Event-Listener gehören in `site.ini [Event] Listeners[]`, die der Kernel korrekt anhängt (Ereignisse, Listener und Filter).

cleanExit() beendet die Anfrage mit einer Exception

Eine View, die ihre eigene Ausgabe sendet (einen Download, JSON), endet mit `eZExecution::cleanExit()`. Unter Velocity beendet das die Anfrage durch eine Exception, die der Server behandelt. **Umschließen Sie es niemals mit** `catch ( Exception $e )` (oder `Throwable`): Das catch verschluckt das Ende der Anfrage, der Kernel läuft weiter, und die Seite wird an Ihre Ausgabe angehängt.

PHP

```php
header( 'Content-Type: application/json; charset=utf-8' );
echo json_encode( $data );
eZExecution::cleanExit();           // not inside a try block that catches Exception
```

Neue Klassen und neue Einstellungen

- Ein Worker behält die Klassen-Map, mit der er gestartet ist: Eine Klasse, die nach dem Start der Worker hinzugefügt wurde, existiert für sie nicht. Generieren Sie die Autoloads neu und starten Sie die Worker neu.
- Eine Einstellung, die eine Klasse nennt (einen Handler, einen Listener, ein `Implementation[]`), bricht Worker, die die Klasse nicht kennen. Stellen Sie zuerst den Code bereit, starten Sie neu, dann die Einstellungen, und sichern Sie mit `class_exists()` ab, wo eine fehlende Klasse keine Seiten brechen darf.
- Einstellungen werden einmal pro Worker gelesen: Leeren Sie nach einer Änderung den INI-Cache (`--clear-tag=ini`) und starten Sie neu.

Eine PHP-Änderung bereitstellen

Shell

```shell
php bin/php/console exp:velocity deploy --dry-run     # what it would do
php bin/php/console exp:velocity deploy               # the usual case
php bin/php/console exp:velocity deploy --kernel      # a kernel class was added or renamed
```

Ein Befehl erledigt die Schritte in der richtigen Reihenfolge und gibt für jeden PASS, FAIL oder

SKIP aus: Autoloads, der INI-Cache, Template- und Override-Caches, das Engine-Archiv (nur wenn sich `kernel/`, `lib/` oder `autoload/` geändert hat; jede Datei wird zuerst geparst), ein Neuladen von PHP-FPM, der Neustart von Velocity und danach die Caches, die gerenderte Ausgabe enthalten (Velocity-Engines: eine PHP-Änderung bereitstellen).

Ihren Code für Velocity testen

1. Rufen Sie die Seite zweimal hintereinander auf demselben Worker auf, mit zwei verschiedenen Benutzern, zwei Siteaccesses, zwei Sprachen. Ausgabe der ersten Anfrage darf niemals in der zweiten erscheinen.
2. Vergleichen Sie mit dem eingebauten Server von PHP (`--engine=php`), der jede Anfrage frisch ausführt: Ein Fehler, den nur Velocity zeigt, ist Zustand, der zwischen Anfragen erhalten bleibt.
3. Sehen Sie im Reiter **Server** der Debug-Leiste nach, welche Engine die Seite gerendert hat.
4. PHPUnit führt alle Tests in einem Prozess aus: Ein Test, der statischen Zustand ändert und nicht zurücksetzt, zeigt dieselbe Art von Fehler (PHPUnit ohne Datenbank).

Verwandte Seiten

- Velocity: Exponential in einem Webserver mit persistenten Workern betreiben
- Mit Velocity bereitstellen
- Hinweise zum Upgrade der Velocity-Engine
- 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.