Statische Analyse mit PHPStan

Wie Exponential PHPStan verwendet - Level 1 über kernel/ und lib/ mit einer Baseline der vorhandenen Befunde, was ausgeschlossen ist und warum, die Ausführung lokal und in der CI, einen Befund lesen, ihn beheben statt...

Source file: `doc/ger-de/hilfe/php-entwickler/statische-analyse-mit-phpstan.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/static-analysis-with-phpstan.md

Diese Seite richtet sich an PHP-Entwickler, die zu Exponential committen oder dasselbe Sicherheitsnetz für eine Erweiterung möchten. PHPStan liest den Code, ohne ihn auszuführen, und meldet Aufrufe nicht existierender Methoden, falsche Argumentanzahlen, undefinierte Variablen und ähnliche Fehler. In Exponential läuft es auf Level 1 mit einer Baseline, sodass nur Befunde fehlschlagen, die Ihre Änderung hinzufügt.

Inhalt

1. Die Konfiguration
2. Ausführen
3. Die Baseline
4. Einen Befund lesen und beheben
5. Was ausgeschlossen ist und warum
6. Ihre Erweiterung analysieren

Die Konfiguration

`phpstan.neon.dist`:

Parameter	Wert	Bedeutung
`includes`	`phpstan-baseline.neon`	die bekannten Befunde, ignoriert
`level`	`1`	undefinierte Variablen, unbekannte Klassen und Methoden, falsche Argumentanzahlen und die Prüfungen von Level 0
`phpVersion`	`80100`	der Code wird als PHP 8.1 analysiert
`paths`	`kernel`, `lib`	was analysiert wird
`excludePaths`	`vendor/*`, `var/*`, `*/tests/*`, `kernel/private/classes/commands/*`, `.../dfs_filter_iterator.php`	siehe unten
`bootstrapFiles`	`autoload.php`	die Klassen-Maps, damit PHPStan die Klassen des Kernels findet
`tmpDir`	`.devtools/phpstan`	der Ergebnis-Cache

Parameter	Wert	Bedeutung
`treatPhpDocTypesAsCertain`	`false`	Docblock-Typen leiten die Analyse, aber ein defensives `is_array()` auf einen als Array dokumentierten Wert ist kein Fehler

Da der Code keine Typdeklarationen hat, stammt das Typwissen von PHPStan aus Docblocks; gute Zeilen `@param` und `@return` verbessern die Analyse (Warum der Code keine Typdeklarationen hat).

Ausführen

Shell

```
make devtools          # once: PHPStan into .devtools/ (composer.dev.json), never into vendor/
make phpstan           # .devtools/vendor/bin/phpstan analyse -c phpstan.neon.dist --memory-limit
=4G
```

Der pre-push-Hook führt denselben Befehl aus, sodass ein Push mit einem neuen Befund abgelehnt wird, sofern Sie nicht `--no-verify` verwenden (Git-Hooks und make-Ziele). In der CI läuft der Job **PHPStan** des Quality-Workflows unter PHP 8.3 und blockiert.

Die Baseline

`phpstan-baseline.neon` listet jeden Befund auf, den der Code hatte, als die Prüfung eingeführt wurde (damals 3033), jeweils mit Meldung, Bezeichner, Anzahl und Pfad. Ein Befund in der Baseline wird ignoriert; ein neuer schlägt fehl. Zwei Regeln:

- **Beheben Sie neue Befunde; nehmen Sie sie nicht in die Baseline auf.** `make phpstan-baseline` akzeptiert alle aktuellen Befunde und ist für die seltene Änderung gedacht, die das Level anhebt oder die Baseline neu ordnet, in einer eigenen Änderung.
- **Verkleinern Sie die Baseline, wenn Sie ohnehin in der Datei sind.** Wenn Sie einen Befund aus der Baseline beheben, passt PHPStan diesen Eintrag nicht mehr an; `reportUnmatchedIgnoredErrors: false` verhindert, dass das fehlschlägt, und eine spätere Neuerzeugung der Baseline entfernt den Eintrag.

Einen Befund lesen und beheben

Ein Befund nennt die Datei, die Zeile, die Meldung und einen Bezeichner:

```
------ ------------------------------------------------------------
 Line   extension/acmereport/classes/acmereportservice.php
------ ------------------------------------------------------------
 42     Call to an undefined method acmeReportItem::fetchByOwner().
        method.notFound
------ ------------------------------------------------------------
```

Häufige Meldung	Übliche Ursache	Lösung
`Variable $x might not be defined`	nur in einem Zweig zugewiesen	initialisieren Sie sie vor dem Zweig

Häufige Meldung	Übliche Ursache	Lösung
`Call to an undefined method`	ein Tippfehler oder eine Methode einer anderen Klasse	verwenden Sie die richtige Methode; prüfen Sie `kernel/classes/`
`... invoked with 3 parameters, 2 required`	die Signatur hat sich geändert	passen Sie den Aufruf an
`Access to an undefined property`	eine dynamische Eigenschaft	deklarieren Sie sie in der Klasse
`Instantiated class X not found`	ein fehlender Eintrag in der Klassen-Map oder ein Tippfehler	generieren Sie die Autoloads neu; korrigieren Sie den Namen

Viele Befunde der Form `Variable $Params might not be defined` in der Baseline stammen aus View-Dateien, die `$Params` von `eZProcess::runFile()` erhalten. Die klassenbasierten Views (`Exponential\Runnable\ModuleView`) nehmen sie stattdessen aus `$scope` und haben das Problem nicht (Modul-Views als Klassen).

Was ausgeschlossen ist und warum

Pfad	Grund (aus `phpstan.neon.dist`)
`vendor/*`, `var/*`	Code von Drittanbietern und generierter Code
`*/tests/*`	Tests werden geprüft, indem man sie ausführt
`kernel/private/classes/commands/*`	Befehlsklassen, die auch globale Hilfsfunktionen deklarieren; zwei davon in einem Analyseprozess zu laden deklariert sie erneut
`kernel/private/classes/clusterfilehandlers/dfsbackends/dfs_filter_iterator.php`	seinen Iterator-Methoden fehlen die vorläufigen Rückgabetypen von PHP 8.1 oder `#[\ReturnTypeWillChange]`, was PHPStan nicht in die Baseline aufnehmen kann

Ihre Erweiterung analysieren

PHPStan ist nicht auf den Kernel beschränkt. Mit installierten Entwicklungswerkzeugen richten Sie es mit einer kleinen eigenen Konfiguration, die den Autoloader des Kernels bootstrapt, auf Ihre Erweiterung:

```
# extension/acmereport/phpstan.neon
parameters:
    level: 1
    paths:
        - .
    excludePaths:
        - tests/*
    bootstrapFiles:
        - ../../autoload.php
```

Shell

```
.devtools/vendor/bin/phpstan analyse -c extension/acmereport/phpstan.neon --memory-limit=2G
```

Generieren Sie zuerst die Autoloads neu (`php bin/php/ezpgenerateautoloads.php -e`), damit Ihre eigenen Klassen bekannt sind. Beginnen Sie wie der Kernel auf Level 1 und heben Sie das Level an, wenn die Befunde verschwunden sind.

Verwandte Seiten

- Qualitätsprüfungen
- Der Coding-Standard (phpcs.xml.dist)
- CI und das Nachstellen einer roten Prüfung
- 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.