Der Coding-Standard (phpcs.xml.dist)

Wie PHP-Code in Exponential geschrieben und geprüft wird - jede Regel aus phpcs.xml.dist mit einem Beispiel, die verbotenen Funktionen, die Prüfung auf PHP-8.0-Kompatibilität, warum nur neue Verstöße fehlschlagen, und…

Source file: `doc/ger-de/hilfe/php-entwickler/der-coding-standard.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/the-coding-standard.md

Diese Seite richtet sich an PHP-Entwickler, die Code für den Kernel oder für Erweiterungen schreiben, die dem Stil des Kernels folgen. Der Stil ist in `phpcs.xml.dist` beschrieben und wird von PHP_CodeSniffer geprüft. Jede Regel wurde vor ihrer Aufnahme am vorhandenen Kernel- und Bibliothekscode gemessen, sodass der Standard beschreibt, wie der Code bereits aussieht; es ist nicht PSR-12 (Coding-Standards).

Inhalt

1. Ein Beispiel im Hausstil
2. Die Regeln
3. Verbotene Funktionen
4. Typdeklarationen und PHP 8.0
5. Nur neue Verstöße schlagen fehl
6. Die Prüfung ausführen
7. Über das hinaus, was das Werkzeug prüft

Ein Beispiel im Hausstil

PHP

```php
<?php
/**
 * File containing the acmeReportService class.
 *
 * @package acmereport
 */

class acmeReportService
{
    /**
     * The totals of the items, by status.
     *
     * @param array $items list of acmeReportItem
     * @return array status => total
     */
    public static function totals( array $items )
    {
        $totals = array();
        foreach ( $items as $item )
        {
            $status = (string)$item->attribute( 'status' );
            if ( !isset( $totals[$status] ) )
            {
                $totals[$status] = 0;
```

```
            }
            $totals[$status] += (int)$item->attribute( 'amount' );
        }
        return $totals;
    }
}
```

Die Regeln

Regel	Richtig	Falsch
Lange Array-Syntax (`Generic.Arrays.DisallowShortArraySyntax`)	`array( 1, 2 )`	`[ 1, 2 ]`
Ein Leerzeichen innerhalb der Klammern von Kontrollstrukturen	`if ( $a )`, `foreach ( $list as $x )`	`if ($a)`
Ein Leerzeichen innerhalb der Klammern von Aufrufen, keines vor (	`foo( $a, $b )`	`foo($a, $b)`, `foo ( $a )`
Ein Leerzeichen innerhalb der Klammern von Deklarationen, = mit Leerzeichen	`function foo( $a, $b = 1 )`	`function foo($a, $b=1)`
Geschweifte Klammer der Funktion in eigener Zeile (`OpeningFunctionBraceBsdAllman`)	`function foo()`, dann { in der nächsten Zeile	`function foo() {`
Kein Leerzeichen nach !	`!$a`	`! $a`
Schlüsselwörter und Konstanten in Kleinbuchstaben	`true`, `false`, `null`, `foreach`	`TRUE`, `NULL`
Vollständiges Öffnungs-Tag	`<?php`	`<?`
Unix-Zeilenenden, Leerzeichen statt Tabulatoren, keine Leerzeichen am Zeilenende		

Die Aufteilung eines Aufrufs über mehrere Zeilen ist frei; nur die Leerzeichen innerhalb der Klammern werden geprüft.

Verbotene Funktionen

`Generic.PHP.ForbiddenFunctions` lehnt Funktionen ab, die PHP entfernt hat oder die Exponential ersetzt:

Verboten	Verwenden Sie
`each`	`foreach`
`create_function`	eine Closure
`split`	`explode` oder `preg_split`
`ereg`, `eregi`	`preg_match`
`ereg_replace`	`preg_replace`
`mysql_query`, `mysql_connect`	eZDB
`sizeof`	`count`
`delete`	`unset`

Auch die Übergabe per Referenz zur Aufrufzeit (`foo( &$a )`) wird abgelehnt.

Typdeklarationen und PHP 8.0

Zwei weitere Teile gehören zum Standard:

- **Keine Typdeklarationen ab PHP 7**, geprüft vom projekteigenen Sniff `bin/phpcs/Exponential/Sniffs/TypeDeclarations/NoTypeDeclarationsSniff.php`: kein `declare( strict_types=1 )`, keine skalaren, Pseudo-, Vereinigungs- oder Schnittmengen-Parametertypen, keine Rückgabe-, Eigenschafts- oder Konstantentypen. Erlaubt sind die Hints, die es schon in PHP 5 gab: `array`, `callable`, `self` und Klassennamen, auch nullbar (`?array`, `?eZINI`). Dateien unter `tests/` dürfen das von PHPUnit verlangte `: void` deklarieren. Die Gründe stehen unter Warum der Code keine Typdeklarationen hat.
- **PHPCompatibility** mit `testVersion 8.0-`: keine Syntax und keine Funktionen, die neuer als PHP 8.0 sind, und keine, die spätere Versionen entfernt haben oder als veraltet markieren (Code schreiben, der unter PHP 8.0 läuft).

Nur neue Verstöße schlagen fehl

Alte Dateien folgen nicht jeder Regel, und das Projekt formatiert alten Code nie aus Stilgründen um. `bin/php/checkcodestyle.php` vergleicht deshalb die Anzahl der Verstöße jeder geänderten Datei mit ihrer vorherigen Version und schlägt nur fehl, wenn Ihre Änderung Verstöße hinzufügt. Den alten Stil der Zeilen, die Sie nicht berührt haben, dürfen Sie stehen lassen; Sie dürfen ihn nicht vermehren.

Die Prüfung ausführen

Shell

```
make devtools                                          # once: PHP_CodeSniffer and PHPCompatibility
 into .devtools/
make phpcs                                             # changed files since origin/main; only new
violations fail
make phpcs BASE=origin/release-branch                  # against another base
php bin/php/checkcodestyle.php --staged                # staged files against HEAD (what the pre-co
mmit hook runs)
.devtools/vendor/bin/phpcs --standard=phpcs.xml.dist extension/acmereport/   # a whole directory,
 all findings
```

In der CI ist der Schritt für den Code-Stil vorerst eine Warnung, kein Fehlschlag; der pre-commit-Hook lehnt einen Commit ab, der Verstöße hinzufügt, sofern Sie nicht `--no-verify` verwenden. Der Sniff hat eigene Tests: `vendor/bin/phpunit --testsuite quality` (Git-Hooks und make-Ziele).

Über das hinaus, was das Werkzeug prüft

Dies sind die Konventionen der Codebasis, die kein Sniff prüft; das tun die Reviewer:

- Eine Klasse pro Datei, Dateiname in Kleinbuchstaben (`acmereportservice.php` für `acmeReportService`).
- Ein Docblock an jeder Klasse und jeder öffentlichen Methode mit Typen bei `@param` und `@return`, da der Code keine Typdeklarationen hat (das Beispiel oben zeigt die Form).
- Neue Bezeichner tragen Ihr Präfix; neuer Projektcode verwendet `exp`; Kernel-Namen `eZ*`

werden nie umbenannt (Klassenbenennung und das Präfix exp).

- `eZDebug::writeError()` und verwandte Methoden mit `__METHOD__` als Label.
- Amerikanisches Englisch in Kommentaren und Meldungen.

Verwandte Seiten

- Coding-Standards
- Qualitätsprüfungen
- Statische Analyse mit PHPStan
- 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.