PHPUnit ohne Datenbank

Wie das PHPUnit-Setup von Exponential für PHP-Entwickler funktioniert - phpunit.xml, tests/bootstrap.php, die Testsuites und ausgeschlossenen Gruppen, PHPUnit 10 bis 13 unter PHP 8.1 bis 8.5, einen Test ohne Datenbank...

Source file: `doc/ger-de/hilfe/php-entwickler/phpunit-ohne-datenbank.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/phpunit-without-a-database.md

Diese Seite richtet sich an PHP-Entwickler, die Unit-Tests für Kernel-Code oder Erweiterungen schreiben. Die meisten Tests von Exponential laufen ohne Datenbank, ohne installierte Site und ohne Webserver, in Sekunden, auf jeder Maschine und in der CI unter PHP 8.1 bis 8.5. Die Seite für Anwendungsentwickler zeigt einen ersten Test (PHPUnit-Tests ohne Datenbank); diese Seite erklärt das Setup darunter.

Inhalt

1. Die Konfiguration
2. Der Bootstrap
3. Suites und Gruppen
4. PHPUnit-Versionen
5. Einen Test schreiben
6. Einstellungen in einem Test
7. Zustand zwischen Tests
8. Nur das ausführen, was Sie geändert haben

Die Konfiguration

`phpunit.xml` im Wurzelverzeichnis setzt:

Einstellung	Wert
`bootstrap`	`tests/bootstrap.php`
Testsuites	`security`, `kernel-classes`, `kernel-content`, `kernel-datatypes`, `kernel-search`, `kernel-private`, `lib`, `mongodb`, `ezjscore`, `expservices`, `ezoe`, `quality`, `design` und weitere
ausgeschlossene Gruppen	`database`, `mail-live`, `mongodb-live`, `network-live`
`source`	`kernel/` und `lib/`
Umgebung	`EZP_TEST_ENVIRONMENT=1`, `memory_limit=1G`

Shell

```
php vendor/bin/phpunit                                    # every suite, without the excluded group
s
php vendor/bin/phpunit --testsuite kernel-classes         # one suite
php vendor/bin/phpunit tests/tests/lib/ezutils/           # one directory
php vendor/bin/phpunit --filter testRefusesNull <file>    # one test
php vendor/bin/phpunit --list-tests                       # what would run
```

Der Bootstrap

`tests/bootstrap.php` lädt den Autoloader von Composer (der über `files` in `composer.json` `autoload.php` lädt), schaltet die Test-Klassen-Map `var/autoload/ezp_tests.php` ein (`ezpAutoloader::enableTests()`, generiert mit `php bin/php/ezpgenerateautoloads.php -s`), bereitet die isolierten Kindprozesse von PHPUnit vor und gibt dem Lauf ein eigenes Audit-Verzeichnis unter `var/tmp`. Er bootet den Kernel **nicht**: kein Siteaccess, keine Datenbankverbindung, keine Sitzung. Code, der nur Klassen und Einstellungen braucht, läuft; Code, der die Datenbank abfragt, nicht.

Suites und Gruppen

- **Suites** gruppieren die Testdateien nach Verzeichnis unter `tests/tests/` (`tests/tests/kernel/classes/`, `tests/tests/lib/` …) und nach Erweiterung.
- **Gruppen** markieren Tests, die etwas außerhalb des Prozesses brauchen. Versehen Sie einen Test mit `@group database` (oder einem Attribut `#[Group('database')]`), wenn er eine Datenbank braucht; der normale Lauf überspringt ihn, und der ältere Runner führt ihn aus (Datenbank- und Live-Tests).

PHPUnit-Versionen

`composer.json` erlaubt `^9.6 || ^10.5 || ^11.5 || ^12.0 || ^13.4`. Die CI löst PHPUnit je Zweig auf: 10.5 unter PHP 8.1, 11.5 unter 8.2 und 8.3, 13 unter 8.4 und 8.5. Ein Test muss unter allen funktionieren:

- Verwenden Sie Attribute oder Annotationen, die die älteste verwendete Version versteht, oder beides.
- Vermeiden Sie Assertions, die in neueren PHPUnit-Releases hinzugekommen sind, sofern Sie nicht auf sie prüfen.
- Template-Methoden (`setUp()`, `tearDown()`) deklarieren `: void`, was PHPUnit verlangt; es ist der einzige Rückgabetyp, den der Coding-Standard erlaubt, und nur unter `tests/`.

Die Hinweise zur Migration stehen unter PHPUnit 10 und PHPUnit 13.

Einen Test schreiben

PHP

```php
<?php
// tests/tests/kernel/classes/acmeReportServiceTest.php, or extension/acmereport/tests/acmeReport
ServiceTest.php
class acmeReportServiceTest extends PHPUnit\Framework\TestCase
{
    public function testTotalsAddAmountsPerStatus()
    {
        $items = array( new acmeReportItem( array( 'status' => 'open', 'amount' => 2 ) ),
                        new acmeReportItem( array( 'status' => 'open', 'amount' => 3 ) ) );
        $this->assertSame( array( 'open' => 5 ), acmeReportService::totals( $items ) );
    }

    public function testTotalsOfNothingIsEmpty()
    {
        $this->assertSame( array(), acmeReportService::totals( array() ) );
    }
}
```

Eine Klasse, die `eZPersistentObject` erweitert, lässt sich wie oben ohne Datenbank aus einem Array konstruieren; nur `store()`, `fetchObject()` und verwandte Methoden brauchen eine. Halten Sie die Logik, die Sie testen möchten, in Methoden, die Werte nehmen und Werte zurückgeben, und die Datenbankaufrufe in schlanken Methoden darum herum.

Einstellungen in einem Test

`tests/toolkit/ezpinihelper.php` setzt INI-Werte im Prozess und stellt sie wieder her:

PHP

```php
public function setUp(): void
{
    ezpINIHelper::setINISetting( 'site.ini', 'DebugSettings', 'DebugOutput', 'disabled' );
}

public function tearDown(): void
{
    ezpINIHelper::restoreINISettings();
}
```

Ein Test, der einen INI-Wert ändert und nicht wiederherstellt, ändert die Einstellungen des nächsten Tests. Das ist eine klassische Ursache für einen Test, der allein besteht und im gesamten Lauf fehlschlägt.

Zustand zwischen Tests

Viele Kernel-Klassen halten statische Caches (Einstellungen, Objekte, den aktuellen Benutzer). PHPUnit führt jeden Test eines Laufs in einem Prozess aus, sodass dieselbe Disziplin gilt wie unter den persistenten Workern von Velocity (Fallen durch statischen Zustand): Setzen Sie zurück, was Sie geändert haben, und verlassen Sie sich nicht auf die Reihenfolge. Wenn zwei Dateien nur gemeinsam fehlschlagen, führen Sie sie zusammen aus, um das Paar zu finden.

Nur das ausführen, was Sie geändert haben

`tests/test-map.json` ordnet Quelldateien den Tests zu, die sie abdecken: von Hand erstellte Einträge (`map`), Einträge, die aus Testdateien generiert werden, die Quellpfade wörtlich nennen (`generated`), und Dateien, die absichtlich kein Test abdeckt (`ignore`). Das schnelle Push-Gate des Teams führt daraus die Tests einer Änderung aus. Wenn Sie einen Test für eine Datei hinzufügen, die er nicht wörtlich nennt, fügen Sie einen `map`-Eintrag hinzu (Qualitätsprüfungen).

Verwandte Seiten

- Standard für Tests und CI
- Tests für Erweiterungen
- Anwendungen testen
- 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.