Debugging mit eZDebug

Die Debug-API des Kernels für PHP-Entwickler - eZDebug::writeNotice, writeWarning, writeError und writeDebug mit __METHOD__ als Label, bedingte Debug-Meldungen mit eZDebugSetting und debug.ini, Zeitpunkte und…

Source file: `doc/ger-de/hilfe/php-entwickler/debugging-mit-ezdebug.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/debugging-with-ezdebug.md

Diese Seite richtet sich an PHP-Entwickler, die sehen müssen, was ihr Code innerhalb einer Anfrage tut. `eZDebug` (`lib/ezutils/classes/ezdebug.php`) sammelt Meldungen, Zeitpunkte und Akkumulatoren, schreibt Warnungen und Fehler in die Logs und zeigt alles in der Debug-Leiste an, wenn die Debug-Ausgabe eingeschaltet ist. Die Leiste selbst ist unter Die Debug-Leiste für PHP-Entwickler beschrieben.

Inhalt

1. Meldungen schreiben
2. Bedingte Debug-Meldungen
3. Zeitpunkte und Akkumulatoren
4. Die Debug-Ausgabe einschalten
5. Wer Debug-Ausgabe erhält
6. Debugging auf der Kommandozeile
7. Andere Werkzeuge
8. Eine Arbeitsreihenfolge

Meldungen schreiben

PHP

```php
eZDebug::writeNotice( 'Imported ' . $count . ' items', __METHOD__ );
eZDebug::writeWarning( 'Item ' . (int)$id . ' had no title; used the id', __METHOD__ );
eZDebug::writeError( 'The feed could not be read: ' . $message, __METHOD__ );
eZDebug::writeDebug( $details, __METHOD__ );
```

Methode	Stufe	Angezeigt und protokolliert
`writeError()`	error	in der Debug-Ausgabe; immer nach `var/log/error.log` geschrieben (das mitgelieferte `AlwaysLog[]=error`)
`writeWarning()`	warning	in der Debug-Ausgabe; nach `var/log/warning.log` geschrieben, wenn die Stufe protokolliert wird
`writeNotice()`	notice	in der Debug-Ausgabe; nur protokolliert, wenn in `AlwaysLog[]` aufgeführt (es füllt Logs schnell)
`writeDebug()`	debug	in der Debug-Ausgabe; nur protokolliert, wenn in `AlwaysLog[]` aufgeführt

`site.ini` `[DebugSettings]` `AlwaysLog[]` listet die Stufen auf, die auch bei ausgeschalteter Debug-Ausgabe protokolliert werden; der mitgelieferte Wert ist `error`. PHP-Deprecations gehen

nach `var/log/deprecated.log` (`DeprecatedLog=enabled`), jede unterschiedliche Meldung einmal pro Anfrage. Übergeben Sie `__METHOD__` als Label: Es nennt die Klasse und die Methode in jeder Zeile, sodass ein Leser des Logs den Code findet. Schreiben Sie niemals Passwörter, Tokens oder personenbezogene Daten in eine Meldung.

Bedingte Debug-Meldungen

Ausführliche Diagnosen, die Sie nur auf Anfrage möchten, verwenden eine **Debug-Bedingung**:

PHP

```php
eZDebugSetting::writeDebug( 'extension-acmereport', $details, __METHOD__ );
```

Die Meldung wird nur geschrieben, wenn die Bedingung in `debug.ini` aktiviert ist:

INI

```ini
# settings/override/debug.ini.append.php (development)
[GeneralCondition]
extension-acmereport=enabled
```

Der Kernel verwendet Bedingungen wie `kernel-content-view`, `kernel-urltranslator` und `workflow-event`; `ezformtoken` verwendet `ezformtoken`. Die Debug-Leiste findet jede Bedingung von `[GeneralCondition]` und zeigt sie als Schalter an.

Zeitpunkte und Akkumulatoren

PHP

```php
eZDebug::addTimingPoint( 'acmereport: before import' );

eZDebug::createAccumulatorGroup( 'acmereport_total', 'Acme report' );
eZDebug::accumulatorStart( 'acmereport_fetch', 'acmereport_total', 'Fetch the feed' );
$feed = acmeReportFeed::fetch( $url );
eZDebug::accumulatorStop( 'acmereport_fetch' );
```

Zeitpunkte markieren Momente der Anfrage; Akkumulatoren summieren die Zeit eines Codestücks über alle seine Aufrufe (und zählen sie). Beide erscheinen im Reiter **Zeiten** der Leiste. Die Datenbanktreiber erfassen ihre Anweisungen als Akkumulatoren, und daher stammen die SQL-Werte der Leiste.

Die Debug-Ausgabe einschalten

Nur auf einer Entwicklungsmaschine (besser in einer `site.ini.dev.append.php`, siehe Einen PHP-Entwicklungs-Checkout einrichten):

INI

```ini
[DebugSettings]
DebugOutput=enabled

[TemplateSettings]
ShowUsedTemplates=enabled
ShowTemplatePathComments=enabled

[DatabaseSettings]
SQLOutput=enabled
```

`ShowTemplatePathComments` schreibt den Template-Pfad als HTML-Kommentar um die Ausgabe jedes Templates (Kommentare mit Template-Pfaden). Schalten Sie die Debug-Ausgabe niemals für Besucher einer Produktions-Site ein: Sie zeigt Interna.

Wer Debug-Ausgabe erhält

Einstellung	Bedeutung
`[DebugSettings] DebugByIP=enabled` mit `DebugIPList[]`	nur diese Adressen oder Netze (IPv4 und IPv6, CIDR, mit optionalem Label und Ablauf)
`DebugByUser=enabled` mit `DebugUserIDList[]`	nur diese Benutzer
`DebugIPList[]=commandline`	auch die Kommandozeile

INI

```ini
DebugIPList[]=203.0.113.7/32 ; Laptop ; expires=2026-10-02T15:00
```

(Debug-Leiste: die IP-Liste).

Debugging auf der Kommandozeile

Jeder `eZScript`-Befehl nimmt `-d` (oder `--debug`) mit den anzuzeigenden Elementen, zum Beispiel `-d all` oder `-d error,warning,timing`:

Shell

```sh
php bin/php/console ext:acmereport:report --month=2026-09 -d all
```

Andere Werkzeuge

- **Xdebug**: funktioniert wie gewohnt unter PHP-FPM und auf der Kommandozeile; `tests/xdebug.ini` ist die Konfiguration, die die PHPUnit-Kindprozesse verwenden. Lassen Sie es aus, wenn Sie Performance messen.
- **Die Logs**: Alles, was ohne Debug-Ausgabe schiefging, steht in `var/log/` (Logs, der Log-Viewer und Anfrage-Logs).
- **Eine Fehlerreferenz**: Fehlerseiten zeigen eine Referenz, die `php bin/php/explogs.php --reference=<reference>` in den Logs findet (Einen Fehler anhand seiner Referenz finden).

Eine Arbeitsreihenfolge

1. Reproduzieren Sie auf einer Entwicklungskopie mit eingeschalteter Debug-Ausgabe.
2. Lesen Sie den Reiter **Meldungen** auf Fehler und Warnungen aus Ihren Dateien.
3. Prüfen Sie den Reiter **Templates** und die Kommentare mit Template-Pfaden: Ist Ihr Code überhaupt der, der läuft?
4. Fügen Sie `eZDebug::writeDebug()` oder eine Bedingung um den verdächtigen Code ein.
5. Vergleichen Sie unter Velocity mit dem eingebauten Server von PHP (`--engine=php`): Wenn der Fehler verschwindet, suchen Sie nach Zustand, der zwischen Anfragen erhalten bleibt (Fallen durch statischen Zustand).

6. Schreiben Sie einen Test, der fehlschlägt, beheben Sie den Fehler und behalten Sie den Test.

Verwandte Seiten

- Referenz zu eZDebug und Logging
- Debugging und Logs (Anwendungsentwickler)
- Verbesserungen der Debug-Ausgabe
- 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.