Fehler und Fehlersuche

Die Meldungen, die die Template-Engine protokolliert - unbekannte Variable, unbekanntes Attribut, unbekannter Operator oder unbekannte Funktion, Parserfehler mit Zeile und Spalte, ein fehlendes Template, Fehlgebrauch…

Source file: `doc/ger-de/hilfe/template-sprache/fehler-und-fehlersuche.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/template-language/errors-and-debugging.md

Die Template-Engine hält eine Seite nie wegen eines Fehlers im Template an. Sie protokolliert einen Fehler oder eine Warnung, gibt für den fehlerhaften Teil nichts (oder den Rest des Texts) aus und macht weiter. Diese Seite zeigt die Meldungen, die sie erzeugt, jede auf einer Entwicklungsinstallation von einem kleinen Template ausgelöst, damit Sie sie in der Debug-Ausgabe wiedererkennen, und die Einstellungen, die helfen, das Template hinter einem Stück HTML zu finden.

Inhalt

1. Wohin die Meldungen gehen
2. Variablen und Attribute
3. Operatoren und Funktionen
4. Parserfehler
5. Fehlende Templates
6. def, set und undef
7. Die Leerraumregel
8. Debug-Einstellungen
9. Verwandte Seiten

Wohin die Meldungen gehen

`eZTemplate::error()` und `eZTemplate::warning()` (`lib/eztemplate/classes/eztemplate.php`) behalten jede Meldung im Fehler- oder Warnungslog der Engine und geben sie mit Template-Datei, Zeile und Spalte an `eZDebug` weiter. Sie sehen sie

- in der Debug-Ausgabe der Seite, wenn `[DebugSettings] DebugOutput=enabled` gesetzt ist und Ihre IP oder Ihr Benutzer zugelassen ist (`DebugByIP`, `DebugByUser`);
- in `var/log/error.log` und `var/log/warning.log`, wenn das Debug-Log für diese Stufe eingeschaltet ist;
- auf der Seite selbst nur mit `template.ini [ControlSettings] DisplayWarnings=enabled` (standardmäßig und auf einer Entwicklungsinstallation ausgeschaltet), was einen Warnkasten für Template-Fehler hinzufügt.

Die Beispiele unten liefen auf einer Entwicklungsinstallation durch die Engine; jedes zeigt die Ausgabe und dann, was die Engine protokolliert hat. Die Dateipfade in den Parsermeldungen sind die Arbeitspfade, in denen die Beispiele lagen.

Variablen und Attribute

Eine nicht vorhandene Variable gibt nichts aus und protokolliert einen Fehler mit dem Namensraum, in dem sie gesucht wurde:

Template

```
before [{$no_such_variable}] after
```

Ausgabe:

```
before [] after
```

Von der Template-Engine protokolliert:

```
ERROR : Unknown template variable 'no_such_variable' in namespace ''
```

Ein Attribut, das ein Objekt oder Array nicht hat: Die Meldung nennt die Klasse (oder die Größe des Arrays) und führt auf, was existiert:

Template

```
{def $list = array( 'a', 'b' )}
[{$list.5}] [{$list.first}]
```

Ausgabe:

```
[] []
```

Von der Template-Engine protokolliert:

```
ERROR : No such attribute for array(2): 5
Choose one of following: 0, 1
ERROR : No such attribute for array(2): first
Choose one of following: 0, 1
```

Bei einem Objekt lautet die Meldung `No such attribute for object(<class>): <name>`; siehe Das Objektmodell in Templates. Nutzen Sie `is_set()`, um ohne Meldung zu prüfen.

Operatoren und Funktionen

Ein auf diesem Siteaccess nicht registrierter Operator protokolliert eine Warnung und **reicht seine Eingabe unverändert durch**:

Template

```
before [{'text'|no_such_operator}] after
```

Ausgabe:

```
before [text] after
```

Von der Template-Engine protokolliert:

```
WARNING : Operator 'no_such_operator' is not registered
```

Ein unbekanntes Funktions-Tag protokolliert eine Warnung, und vom Tag wird nichts ausgegeben. Beachten Sie, dass auch der Text nach dem Tag in derselben Zeile nicht ausgegeben wurde:

Template

```
before {no_such_function param=1} after
```

Ausgabe:

```
before
```

Von der Template-Engine protokolliert:

```
WARNING : Function "no_such_function" is not registered
```

Beides geschieht, wenn eine Erweiterung, die den Operator oder die Funktion registriert, auf dem Siteaccess nicht aktiv ist oder ihre Autoload-Registrierung fehlt (siehe Template-Operatoren schreiben).

Parserfehler

Ein Tag ohne schließende Klammer: Der Parser protokolliert, wo das Tag beginnt (`file:line[column]`), und gibt den Rest des Templates als Text aus:

Template

```
line one
{def $a = 1
line three
```

Ausgabe:

```
line one
{def $a = 1
line three
```

Von der Template-Engine protokolliert:

```
ERROR : parser error @ tm-errors/04-unterminated-tag.tpl:1[0]
Unterminated tag, needs a } to end the tag.
{def $a = 1
line three
```

Überzählige Zeichen in den Parametern einer Funktion: Der Parser protokolliert, was er erwartet hat; der verstandene Teil läuft trotzdem (`$a` wurde auf 1 gesetzt):

Template

```
{def $a = 1 2}
[{$a}]
```

Ausgabe:

```
[1]
```

Von der Template-Engine protokolliert:

```
ERROR def: parser error @ tm-errors/05-extra-characters.tpl:1
($) expected at [2]
```

Ein Block, der nie geschlossen wird, wird überhaupt nicht gemeldet: Der Parser behandelt den Rest der Datei als Rumpf des Blocks.

Template

```
{if true()}
open if, never closed
```

Ausgabe:

```
open if, never closed
```

Fehlende Templates

Ein `include` (oder eine Ansichtsfunktion), dessen Template in keiner Design-Basis gefunden wird, protokolliert eine Warnung und gibt nichts aus:

Template

```
before [{include uri='design:no/such/template.tpl'}] after
```

Ausgabe:

```
before [] after
```

Von der Template-Engine protokolliert:

```
WARNING : No template could be loaded for "no/such/template.tpl" using resource "design"
```

Zur Suchreihenfolge hinter `could not be loaded` siehe Template-Ressourcen und Overrides.

def, set und undef

`def` auf eine vorhandene Variable warnt und weist trotzdem zu; `set` auf eine nicht vorhandene Variable warnt und tut nichts; `undef` auf eine nicht mit `def` angelegte Variable warnt:

Template

```
{def $a = 1}
{def $a = 2}
{set $b = 3}
{undef $c}
[{$a}]
```

Ausgabe:

```
[2]
```

Von der Template-Engine protokolliert:

```
WARNING def: Variable 'a' is already defined.
WARNING set: Variable 'b' doesn't exist, cannot set
WARNING undef: Variable 'c' is not defined with {def}.
```

Die Regeln stehen in `lib/eztemplate/classes/eztemplatedeffunction.php` und `eztemplatesetfunction.php`; die Seite für Designer ist Variablen und def.

Die Leerraumregel

Viele Fragen nach einem „fehlenden Zeilenumbruch“ kommen von einer Regel des Parsers (`eZTemplateMultiPassParser::parseWhitespaceRemoval()`, `lib/eztemplate/classes/eztemplatemultipassparser.php`): **Ein Zeilenumbruch direkt vor und einer direkt nach jedem Funktions-Tag** (`{if}`, `{/if}`, `{foreach}`, `{def ...}`, `{include ...}` und den übrigen) wird entfernt. Ausgabe-Tags wie `{$x}` oder `{$x|wash}` behalten die Zeilenumbrüche um sich. Die Folgen:

- Ein Funktions-Tag allein in seiner Zeile hinterlässt keine Leerzeile, was Templates lesbar macht.
- Text zwischen zwei Funktions-Tags in getrennten Zeilen verliert beide Zeilenumbrüche. Um einen Zeilenumbruch zwischen `{/if}` und dem nächsten `{if}` zu behalten, lassen Sie eine Leerzeile dazwischen; zwei Leerzeilen behalten zwei.
- Eine Zeile, die mit einem Funktions-Tag endet, und die Zeile danach werden verbunden.

Die Beispiele von Data Map und Inhalt nutzen genau deshalb Leerzeilen.

Debug-Einstellungen

Alle in `site.ini [TemplateSettings]`, wenn nicht anders vermerkt; Werte auf dem Siteaccess `site` einer Entwicklungsinstallation:

Template

```
Debug:                    {ezini( 'TemplateSettings', 'Debug' )}
ShowXHTMLCode:            {ezini( 'TemplateSettings', 'ShowXHTMLCode' )}
ShowUsedTemplates:        {ezini( 'TemplateSettings', 'ShowUsedTemplates' )}
ShowTemplatePathComments: {ezini( 'TemplateSettings', 'ShowTemplatePathComments' )}
DebugOutput:              {ezini( 'DebugSettings', 'DebugOutput' )}
DisplayWarnings:          {ezini( 'ControlSettings', 'DisplayWarnings', 'template.ini' )}
```

Ausgabe:

```
Debug:                    disabled
ShowXHTMLCode:            enabled
ShowUsedTemplates:        disabled
ShowTemplatePathComments: disabled
DebugOutput:              enabled
DisplayWarnings:          disabled
```

Einstellung	Wirkung
Debug	fügt Kommentare `<!-- START: including template: <file> (<uri>) -->` und STOP um jedes gelesene Template ein (nicht um ein Pagelayout, das mit `<!DOCTYPE` beginnt) und protokolliert, welche Dateien geladen werden

Einstellung	Wirkung
`ShowXHTMLCode`	gibt bei eingeschaltetem Debug den Template-Pfad zusätzlich als sichtbaren Text in der Seite aus
`ShowUsedTemplates`	zeichnet jedes genutzte Template auf; die Debug-Ausgabe (und die Template-Liste der Debug-Leiste) zeigt sie. Braucht `DebugOutput`
`ShowTemplatePathComments`	schließt jedes gerenderte Template in Kommentare `<!-- START <path> -->` / `<!-- STOP <path> -->` ein, mit oder ohne Compiler; nur für die Entwicklung, nie auf Siteaccesses, die JSON, XML oder RSS ausliefern
`ShowMethodDebug`	protokolliert Beginn und Ende jedes Aufrufs einer Funktion oder eines Operators (nur interpretierte Templates)
`template.ini [ControlSettings] DisplayWarnings`	zeigt Template-Fehler als Warnkasten auf der Seite

Kommentare von `Debug` werden beim Lesen eines Templates eingefügt; bereits vorhandene kompilierte Templates bekommen sie also erst, wenn der Template-Cache geleert ist. `ShowTemplatePathComments` funktioniert mit kompilierten Templates so, wie sie sind.

Um auf einer laufenden Website die Datei hinter einem Stück HTML zu finden, ist `ShowTemplatePathComments` meist der schnellste Weg: Schalten Sie es in einer Entwicklungsinstallation für einen Siteaccess ein, sehen Sie sich den Quelltext der Seite an und suchen Sie nach dem HTML. Die Seite Fehlersuche in Templates geht das durch.

Verwandte Seiten

- Fehlersuche in Templates
- Template-Pfadkommentare
- Häufige Fehler in Templates
- Der Template-Compiler
- Fehlersuche und Logs
- 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.