Inhalte aus PHP

Für CMS-Entwickler, die Inhalte in PHP-Skripten, Importern und Integrationen mit der echten Kernel-API lesen und schreiben. Die Seite behandelt das Ausführen eines Skripts mit geladenem Kernel, das Abrufen von Knoten…

Source file: `doc/ger-de/hilfe/cms-entwickler/inhalte-aus-php.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/cms-developers/content-from-php.md

Diese Seite richtet sich an Entwickler, die Inhalte in PHP brauchen: einen Importer, einen nächtlichen Abgleich mit einem anderen System, einen Bericht, ein eigenes Modul. Der Kernel hat für jedes davon einen unterstützten Weg, und ihn zu verwenden hält Versionen, Workflows, Suche und Caches korrekt. Schreiben Sie nie direkt in die Inhaltstabellen.

Inhalt

1. Ein Skript mit geladenem Kernel ausführen
2. Einen Knoten und sein Objekt lesen
3. Listen und Teilbäume
4. Attributwerte lesen
5. Inhalte anlegen
6. Inhalte aktualisieren
7. Der Benutzer, unter dem ein Skript läuft
8. Verschieben, Verbergen und Entfernen
9. Große Läufe

Ein Skript mit geladenem Kernel ausführen

Am schnellsten führen Sie eine beliebige PHP-Datei mit erledigtem Bootstrap aus, vom Hauptverzeichnis der Installation:

Shell

```
php bin/php/ezexec.php extension/exp_mysite/scripts/report.php
```

Für ein Skript, das Sie behalten, mit Optionen, Hilfe und Fortschrittsausgabe, schreiben Sie ein `eZScript`-Skript oder einen Konsolenbefehl in Ihrer Erweiterung (eZScript und eZCLI, CLI-Skripte und Konsolenbefehle). Die Lektion Ein Skript mit geladenem Kernel führt durch ein erstes.

Einen Knoten und sein Objekt lesen

PHP

```php
<?php
$cli = eZCLI::instance();
$node = eZContentObjectTreeNode::fetch( 2 );
if ( !$node instanceof eZContentObjectTreeNode )
{
    $cli->error( 'Node 2 was not found' );
    return;
}
$object = $node->object();
$cli->output( $node->attribute( 'name' ) . ' (' . $node->attribute( 'class_identifier' ) . ')' );
$cli->output( 'Object ' . $object->attribute( 'id' ) . ', main node ' . $object->attribute( 'main
_node_id' ) );

$imported = eZContentObject::fetchByRemoteID( 'exp-import-event-42' );
```

`eZContentObjectTreeNode::fetch()` nimmt eine Knoten-ID; `eZContentObjectTreeNode::fetchByRemoteID()`, `fetchByURLPath()` und `eZContentObject::fetch()` / `fetchByRemoteID()` finden Inhalte über andere Schlüssel (`kernel/classes/ezcontentobjecttreenode.php`, `kernel/classes/ezcontentobject.php`).

Listen und Teilbäume

`subTreeByNodeID()` ist die PHP-Seite der Fetches `list` und `tree`, mit den Parametern in CamelCase:

PHP

```php
<?php
$params = array( 'Depth' => 1, 'DepthOperator' => 'eq',
                 'ClassFilterType' => 'include', 'ClassFilterArray' => array( 'exp_event' ),
                 'SortBy' => array( 'published', false ),
                 'Offset' => 0, 'Limit' => 20, 'LoadDataMap' => true );
$events = eZContentObjectTreeNode::subTreeByNodeID( $params, $eventsFolderNodeID );
$total = eZContentObjectTreeNode::subTreeCountByNodeID( $params, $eventsFolderNodeID );
```

Weitere Parameter: `AttributeFilter`, `ExtendedAttributeFilter`, `MainNodeOnly`, `IgnoreVisibility`, `Language`, `OnlyTranslated`, `Limitation`, `AsObject`. Zugriffsrichtlinien und Sichtbarkeit gelten wie für den aktuellen Benutzer, sofern Sie sie nicht abschalten; `'Limitation' => array()` überspringt die Zugriffsprüfung und ist nur für Code gedacht, der den Zugriff selbst prüft (eZContentObjectTreeNode).

Attributwerte lesen

PHP

```php
<?php
$dataMap = $node->dataMap();                // attribute identifier => eZContentObjectAttribute
if ( isset( $dataMap['title'] ) && $dataMap['title']->hasContent() )
{
    $title = $dataMap['title']->content();   // the datatype's value
    $asString = $dataMap['title']->toString(); // the same as a string, in the import format
}
```

`content()` gibt zurück, was der Datentyp Templates als `.content` liefert (eine Zeichenkette, ein `eZDateTime`, einen Bild-Handler, ein Array mit `relation_list` …). `toString()` gibt das Format zurück, das `fromString()` liest; das brauchen Sie für Exporte (Die eZDataType-API).

Inhalte anlegen

`eZContentFunctions::createAndPublishObject()` (`kernel/classes/ezcontentfunctions.php`) legt das Objekt, seine erste Version und seinen Knoten an und führt die Veröffentlichungsoperation aus:

PHP

```php
<?php
$object = eZContentFunctions::createAndPublishObject( array(
    'class_identifier' => 'exp_event',
    'parent_node_id'   => $eventsFolderNodeID,
    'remote_id'        => 'exp-import-event-42',
    'attributes'       => array(
        'title'      => 'Opening of the new hall',
        'intro'      => 'Doors open at ten.',
        'start_date' => (string)mktime( 10, 0, 0, 11, 2, 2026 ),
    ),
) );
if ( !$object instanceof eZContentObject )
{
    eZDebug::writeError( 'Could not create the event', __METHOD__ );
}
```

Werte sind Zeichenketten im `fromString()`-Format jedes Datentyps: Text, wie er ist, `1/0` für ein Kontrollkästchen, ein Zeitstempel für Datumsangaben, ein Dateipfad für Bilder und Dateien (mit `storage_dir`), das gespeicherte XML für Rich Text. Setzen Sie bei importierten Inhalten immer `remote_id`, damit der nächste Lauf aktualisiert statt zu verdoppeln (Inhalte aus PHP anlegen, veröffentlichen und ändern).

Inhalte aktualisieren

PHP

```php
<?php
$object = eZContentObject::fetchByRemoteID( 'exp-import-event-42' );
if ( $object instanceof eZContentObject )
{
    eZContentFunctions::updateAndPublishObject( $object, array(
        'attributes' => array( 'intro' => 'Doors open at eleven.' ),
    ) );
}
```

Nur die Attribute, die Sie übergeben, ändern sich. `language` fügt eine Übersetzung desselben Objekts hinzu oder aktualisiert sie (Übersetzungen und Sprachen). Jeder Aufruf legt eine neue Version an und veröffentlicht sie, mit ihren Workflows und Benachrichtigungen; übergeben Sie `false` als drittes Argument, um ohne Benachrichtigungs-Events zu veröffentlichen.

Der Benutzer, unter dem ein Skript läuft

Ein Kommandozeilenskript läuft als anonymer Benutzer, sofern es keinen setzt, und der anonyme Benutzer darf meist keine Inhalte anlegen. Verwenden Sie ein eigenes Importkonto mit genau den Rechten, die der Import braucht, keinen Administrator:

PHP

```php
<?php
$user = eZUser::fetchByName( 'exp_importer' );
if ( $user instanceof eZUser )
{
    eZUser::setCurrentlyLoggedInUser( $user, $user->attribute( 'contentobject_id' ) );
}
```

Die angelegten Objekte erhalten diesen Benutzer als Eigentümer.

Verschieben, Verbergen und Entfernen

Aufgabe	Aufruf
Einen Knoten verschieben	`eZContentObjectTreeNodeOperations::move( $nodeID, $newParentNodeID )`
Einen Teilbaum verbergen oder anzeigen	`eZContentObjectTreeNode::hideSubTree( $node )`, `unhideSubTree( $node )`
Knoten entfernen (standardmäßig in den Papierkorb)	`eZContentObjectTreeNode::removeSubtrees( $nodeIDs, $moveToTrash = true, $infoOnly = false )`

`removeSubtrees()` mit `$infoOnly = true` meldet, was entfernt würde: Verwenden Sie es für einen Probelauf.

Große Läufe

- Leeren Sie zwischendurch den Objekt-Cache im Speicher (`eZContentObject::clearCache()`), sonst wächst der Speicher mit jedem Objekt.
- Bieten Sie ein `--dry-run` an, das meldet, was angelegt oder geändert würde.
- Führen Sie es zuerst an einer Kopie aus, dann live in einer ruhigen Stunde; große Änderungen können auch als Inhaltsjobs im Hintergrund laufen (Inhaltsjobs).
- Für die vorhandenen Importwerkzeuge lesen Sie Inhalte importieren, bevor Sie Code schreiben.

Verwandte Seiten

- Lektion 4.3: Inhalte aus PHP lesen und schreiben
- eZContentObject
- Inhalte in Templates abrufen
- 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.