Einen Datentyp schreiben

Die PHP-Seite eines eigenen Datentyps - die Methoden von eZDataType, gruppiert danach, wann der Kernel sie aufruft, die Registrierung mit eZDataType::register(), die Speicherung in data_text, data_int und data_float...

Source file: `doc/ger-de/hilfe/php-entwickler/einen-datentyp-schreiben.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/write-a-datatype.md

Diese Seite richtet sich an PHP-Entwickler, die einen neuen Feldtyp für Inhaltsklassen schreiben. Ein Datentyp ist eine Klasse, die `eZDataType` (`kernel/classes/ezdatatype.php`) erweitert, plus Bearbeitungs- und Ansichts-Templates. Das vollständige funktionierende Beispiel mit seinen Templates und seiner Registrierung finden Sie unter Einen eigenen Datentyp schreiben; diese Seite ordnet die PHP-Methoden den Momenten zu, in denen der Kernel sie aufruft, damit Sie wissen, welche Sie überschreiben müssen.

Inhalt

1. Registrierung
2. Die Methoden nach Moment
3. Speicherung
4. String-, Paket- und Suchformen
5. Einstellungen des Klassenattributs
6. PHP-Punkte, auf die Sie achten müssen
7. Einen Datentyp testen

Registrierung

PHP

```php
class acmeColorType extends eZDataType
{
    const DATA_TYPE_STRING = 'acmecolor';

    public function __construct()
    {
        parent::__construct( self::DATA_TYPE_STRING, ezpI18n::tr( 'extension/acmereport', 'Color'
 ),
                             array( 'serialize_supported' => true ) );
    }
    // ... the methods below
}

eZDataType::register( acmeColorType::DATA_TYPE_STRING, 'acmeColorType' );
```

Die Datei ist `extension/acmereport/datatypes/acmecolor/acmecolortype.php`, gefunden über `content.ini`:

INI

```ini
<?php /* #?ini charset="utf-8"?
[DataTypeSettings]
ExtensionDirectories[]=acmereport
AvailableDataTypes[]=acmecolor
*/ ?>
```

Der Typ-String wird in jedem Attribut dieses Typs gespeichert, für immer: Wählen Sie ihn einmal, mit Präfix, in Kleinbuchstaben.

Die Methoden nach Moment

Moment	Methoden
ein Redakteur sendet das Bearbeitungsformular ab	`validateObjectAttributeHTTPInput()`, dann `fetchObjectAttributeHTTPInput()`; `customObjectAttributeHTTPAction()` für zusätzliche Schaltflächen
das Attribut wird gespeichert	`storeObjectAttribute()`
eine neue Version oder Übersetzung wird angelegt	`initializeObjectAttribute()`, `postInitializeObjectAttribute()`
das Objekt wird veröffentlicht	`onPublish()`
Templates lesen den Wert	`objectAttributeContent()` (`$attribute.content`), `hasObjectAttributeContent()` (`$attribute.has_content`), `title()`
der Suchindex wird aufgebaut	`isIndexable()`, `metaData()`
Listen werden nach dem Attribut sortiert	`sortKey()`, `sortKeyType()` (`'string'` oder `'int'`)
eine Version wird entfernt oder das Objekt kommt in den Papierkorb	`deleteStoredObjectAttribute()`, `trashStoredObjectAttribute()`, `restoreTrashedObjectAttribute()`
Export und Import	`toString()`, `fromString()`, `serializeContentObjectAttribute()`, `unserializeContentObjectAttribute()`
die Klasse wird bearbeitet	`validateClassAttributeHTTPInput()`, `fetchClassAttributeHTTPInput()`, `initializeClassAttribute()`, `storeClassAttribute()`

`validate...` gibt `eZInputValidator::STATE_ACCEPTED`, `STATE_INVALID` (nach `setValidationError()`) oder `STATE_INTERMEDIATE` zurück. `fetch...` gibt `true` zurück, wenn es einen Wert gespeichert hat. Der Lebenszyklus im Detail: Der Lebenszyklus eines Datentyps.

Speicherung

Wo	Für
`data_text`, `data_int`, `data_float` von `ezcontentobject_attribute`	einfache Werte, einer pro Attributversion und Sprache
`data_text1` ... `data_int1` ... von `ezcontentclass_attribute`	Einstellungen des Klassenattributs
eine eigene Tabelle, verschlüsselt nach Attribut-ID und Version	strukturierte Werte; entfernen Sie Zeilen in `deleteStoredObjectAttribute()`

Jede Version und Übersetzung hat ihre eigene Attributzeile, sodass ein anderswo gespeicherter Wert nach Attribut-ID **und** Version geschlüsselt und in `initializeObjectAttribute()` kopiert werden muss, wenn eine neue Version von einer alten ausgeht.

String-, Paket- und Suchformen

- `toString()` und `fromString()` sind die Klartextform, die von Importen, der REST-API und Skripten verwendet wird; sorgen Sie dafür, dass sie verlustfrei hin und zurück funktionieren: `fromString( toString( x ) )` ergibt `x`.
- `serializeContentObjectAttribute()` und `unserializeContentObjectAttribute()` schreiben und lesen den Wert in Inhaltspaketen; setzen Sie `'serialize_supported' => true` im Konstruktor, wenn Sie sie implementieren.
- `metaData()` ist der Text, den die Suchmaschine indiziert; geben Sie die Wörter zurück, nach denen ein Besucher suchen würde.

Einstellungen des Klassenattributs

Ein Datentyp mit Optionen pro Klassenattribut (eine Standardfarbe, eine Liste erlaubter Werte) liest sie in `fetchClassAttributeHTTPInput()` aus dem Bearbeitungsformular der Klasse, speichert sie in `data_text1` und verwandten Spalten und zeigt sie in seinen Klassen-Templates `class/datatype/edit/<type>.tpl` und `class/datatype/view/<type>.tpl`.

PHP-Punkte, auf die Sie achten müssen

- **Übernehmen Sie die Signaturen der Elternklasse exakt.** Ein Parameter weniger oder eine Referenzmarkierung, die die Elternklasse nicht hat, ist ein fataler Fehler beim Laden der Klasse, und jede Seite, die Inhalte berührt, bricht (Hinweise zur PHP-8-Migration für Ihren Code).
- **Deklarieren Sie die Eigenschaften**, die Ihre Unterklasse verwendet; PHP 8.2 markiert dynamische Eigenschaften als veraltet.
- `null` **aus der Datenbank.** `data_text` kann `null` sein; casten Sie vor String-Funktionen (`(string)`).
- **Maskieren Sie in Templates**, nicht in `objectAttributeContent()`: Der Inhalt sind Daten, `|wash` macht sie in HTML sicher.

Einen Datentyp testen

1. Ohne Datenbank: Hin- und Rückwege von `toString()`/`fromString()` und die Validierungsregeln in einer separaten Validator-Klasse (PHPUnit ohne Datenbank).
2. Auf einer Installation: Fügen Sie einer Testklasse ein Attribut hinzu; legen Sie Objekte an, bearbeiten, übersetzen und kopieren Sie sie, verschieben Sie sie in den Papierkorb, stellen Sie sie wieder her und entfernen Sie sie; exportieren und importieren Sie ein Paket.
3. Prüfen Sie den Suchindex und die Sortierung, falls Sie sie implementiert haben.

Verwandte Seiten

- Datentypen (Erweiterungsentwickler)
- eZDataType-API
- Das Muster Datentyp
- 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.