

Contao Konferenz 2014  
**Erweiterungen  
entwickeln für Contao  
leicht gemacht**

von Stefan Heimes und Tristan Lins



# Tristan "tril" Lins



Softwareentwickler / Geschäftsführer  
bei bit3 UG

Hauptentwickler von Avisota, Theme+,  
dem Contao Composer Client, ...

[www.facebook.com/tristan.lins](http://www.facebook.com/tristan.lins)

[twitter.com/TristanLins](https://twitter.com/TristanLins)

[plus.google.com/+TristanLins](https://plus.google.com/+TristanLins)

[t.lins@c-c-a.org](mailto:t.lins@c-c-a.org)



# Stefan “Chibineko” Heimes



Softwareentwickler bei  
MEN AT WORK Werbeagentur GmbH

Hauptentwickler von syncCto und den  
Language Tools. Mitentwickler von  
MetaModels und DC\_General, ...

[s.heimes@c-c-a.org](mailto:s.heimes@c-c-a.org)



# Modulentwicklung - Themen

- Ordnerstrukturen
- Wichtige Dateien
- Data Container Array
- Klassen / ClassLoader
- Framework
- Pro-Tipps
- Diskussion



# Ordnerstrukturen



# Ordnerstrukturen

- ↳ `system/modules/my_module`
  - ↳ `config`
  - ↳ `dca`
  - ↳ `languages`
  - ↳ `templates`
- ↳ `src|classes|elements|modules|models`
- ↳ `vendor`



# Wichtige Dateien



# config/autoload.php

```
1. <?php
2.
3. ClassLoader::addNamespaces(array(
4.     'My',
5. ));
6.
7. ClassLoader::addClasses(array(
8.     'My\Element' => 'system/modules/my_module/src/My/Element.php',
9.     'My\Module'  => 'system/modules/my_module/src/My/Module.php',
10. ));
11.
12. TemplateLoader::addFiles(array(
13.     'ce_my_element' => 'system/modules/my_module/templates/elements',
14.     'mod_my_module' => 'system/modules/my_module/templates/modules',
15. ));
```





# config/autoload.ini

1. `;;`
2. `; List modules which are required to be loaded beforehand`
3. `;;`
4. `requires[] = "core"`



# config/config.php

```
1. <?php
2.  /** Back end modules */
3.  $GLOBALS['BE_MOD']['my_category']['my_menu_item'] = array('tables' => array('tl_my_table'));
4.  /** Front end modules */
5.  $GLOBALS['FE_MOD']['my_category']['my_frontend_module'] = 'My\Module';
6.  /** Content elements */
7.  $GLOBALS['TL_CTE']['my_category']['my_content_element'] = 'My\Element';
8.  /** Back end form fields */
9.  $GLOBALS['BE_FFL']['my_field'] = 'My\BackEndFormField';
10. /** Front end form fields */
11. $GLOBALS['TL_FFL']['my_field'] = 'My\FrontendFormField';
12. /** Hooks */
13. $GLOBALS['TL_HOOKS']['loadLanguageFile']['my_extension'] = array('My\Hooks', 'loadLanguageFile');
14. /** Models */
15. $GLOBALS['TL_MODELS']['tl_my_table'] = 'My\TableModel';
```



# config/config.php

```
1.  <?php
2.
3.  // Version der eigenen Extension registrieren
4.  define('MY_EXTENSION_VERSION', '1.2.3');
5.
6.  // Eigene Konfigurationsvariablen mit Standardwerten befüllen
7.  // (kann in der Localconfig.php überschrieben werden)
8.  $GLOBALS['TL_CONFIG']['myConfigEntry'] = 'my default value';
9.
10. // Eigene Register-Arrays definieren
11. // KEIN ... => array() ... verwenden!
12. if (!isset($GLOBALS['MY_EXTENSION_REGISTER'])) {
13.     $GLOBALS['MY_EXTENSION_REGISTER'] = array();
14. }
15. $GLOBALS['MY_EXTENSION_REGISTER']['tl_my_item'] = 'something to register';
```



## dca/\*

- Beinhaltet die [Data Container Array](#) Konfigurationen
- Datei muss exakt dem Tabellennamen entsprechen
- Dateien/Tabellen müssen mit tl\_ beginnen
- Bspw: tl\_my\_table



# languages/<sprache>/\*

- <sprache> := Sprach- oder Regionskürzel  
(de oder de\_AT)
- Beinhaltet Übersetzungen
- Kann in PHP oder [XLIFF](#) definiert werden
- Vorteile von XLIFF:
  - Tools für Übersetzungen existieren für alle Systeme
  - Bspw. [Transifex](#)
- Vorteile von PHP:
  - Funktionieren unter Contao <3.1



# languages/<sprache>/\*

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <xliff version="1.1">
3.   <file datatype="php"
      original="system/modules/my_extension/
              languages/en/modules.php"
      source-language="en">
4.     <body>
5.       <trans-unit id="MOD.my_extension">
6.         <source>My extension</source>
7.       </trans-unit>
8.     </body>
9.   </file>
10. </xliff>
```

```
1. <?php
2.
3. $GLOBALS['TL_LANG']['MOD']['my_extension'] =
      'My extension';
```



# templates/\*

- Beinhaltet die Template Dateien
- Templates müssen auf \*.html5 und \*.xhtml enden
  - \*.html5 muss vorhanden sein
  - \*.xhtml sollte vorhanden sein  
(sonst ist keine xhtml Ausgabe möglich)
- Hinweis: Templates müssen in autoload.php registriert werden



# Data Container Array





# Data Container Array

- Dient der Konfiguration des *Data Container*
- Der Data Container stellt (nahezu) alle Eingabemasken im Backend zur Verfügung  
→ bspw. Bearbeiten einer Seite, eines Artikels oder von Inhalten
- Seit Contao 3 auch zur Konfiguration der Datenbank  
→ DCA hat schon immer die Datenbank abgebildet



# DCA Syntax

```
1. <?php
2.
3. $GLOBALS['TL_DCA']['tl_my_table'] = array(
4.
5.     'config' => array( ... ),
6.
7.     'list' => array( ... ),
8.
9.     'palettes' => array( ... ),
10.    'subpalettes' => array( ... ),
11.
12.    'fields' => array( ... ),
13.
14. );
```



# DCA Syntax :: \$GLOBALS[...]['config']

- \$GLOBALS[...]['config']

Abschnitt ist für die allgemeine Konfiguration des Data Container / der Tabelle, bspw.

- Verwendeter Treiber (i.d.R. DC\_Table)
- Eltern- / Kind-Tabelle
- Verfügbare Modi (nicht editierbar / löscherbar / sortierbar / ...)
- Callbacks beim laden / speichern / löschen /... eines Datensatzes
- SQL Indizes
- ...



# DCA Syntax :: \$GLOBALS[...]['config']

```
1. <?php
2. $GLOBALS['TL_DCA']['tl_my_table'] ...
3.
4.     'config' => array(
5.         'label'          => $GLOBALS['TL_LANG']['tl_my_table']['containerHeadline'],
6.         'dataContainer' => 'Table',
7.         'sql' => array(
8.             'keys' => array(
9.                 'id'    => 'primary',
10.                'alias' => 'index',
11.                'email' => 'unique',
12.            )
13.        )
14.    ),
15.
16. ...
```



# DCA Syntax :: \$GLOBALS[...]['list']

- `$GLOBALS['...']['list']`

Abschnitt für die Konfiguration der Auflistung von Datensätzen, bspw.

- Gruppierung (nach Name / Daten / ...)
- Reihenfolge (Aufsteigend / Absteigend)
- Angezeigte Felder des Datensatzes (Label)
- Buttons / Links (Operations)
- ...



# DCA Syntax :: \$GLOBALS[...]['list']

```
1. <?php
2. $GLOBALS['TL_DCA']['tl_my_table'] ...
3.
4.     'list' => array(
5.         'sorting' => array(
6.             'mode' => 5,
7.         ),
8.         'label' => array(
9.             'fields'      => array('title'),
10.            'format'      => '%s',
11.        ),
12.        'global_operations' => array( ... ),
13.        'operations'      => array( ... )
14.    ),
15.
16. ...
```

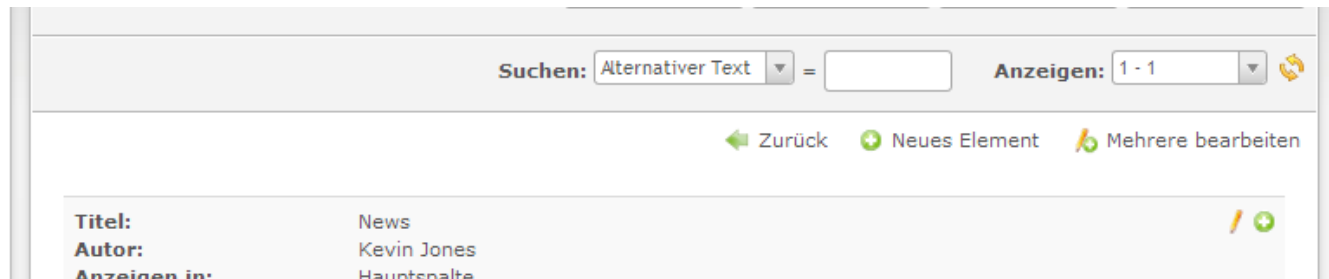


# DCA Syntax :: \$GLOBALS[...]['global\_operations']

- `$GLOBALS['...']['global_operations']`

Abschnitt für die Konfiguration der globalen Aktionen (werden rechts oben angezeigt), bspw.

- zurück zur vorherigen Ansicht
- Mehrere bearbeiten



The screenshot shows a web application interface with a search bar and action buttons. The search bar contains the text "Alternativer Text" and a dropdown menu. To the right of the search bar is a field with an equals sign and another empty field. Further right is a "Anzeigen:" label followed by a dropdown menu showing "1 - 1" and a refresh icon. Below the search bar are three buttons: "Zurück" with a left arrow, "Neues Element" with a plus sign, and "Mehrere bearbeiten" with a group of three people icon. At the bottom, there is a table with the following data:

|                     |             |     |
|---------------------|-------------|-----|
| <b>Titel:</b>       | News        | / + |
| <b>Autor:</b>       | Kevin Jones |     |
| <b>Anzeigen in:</b> | Hauptsalte  |     |



# DCA Syntax :: \$GLOBALS[...]['global\_operations']

```
1. <?php
2. $GLOBALS['TL_DCA']['tl_my_table']['list'] ...
3.
4.     'global_operations' => array(
5.         'all' => array(
6.             'label'      => &$GLOBALS['TL_LANG']['MSC']['all'],
7.             'href'      => 'act=select',
8.             'class'     => 'header_edit_all',
9.             'attributes' => 'onclick="Backend.getScrollOffset()" accesskey="e"'
10.        ),
11.        ...
12.    ),
13.
14. ...
```



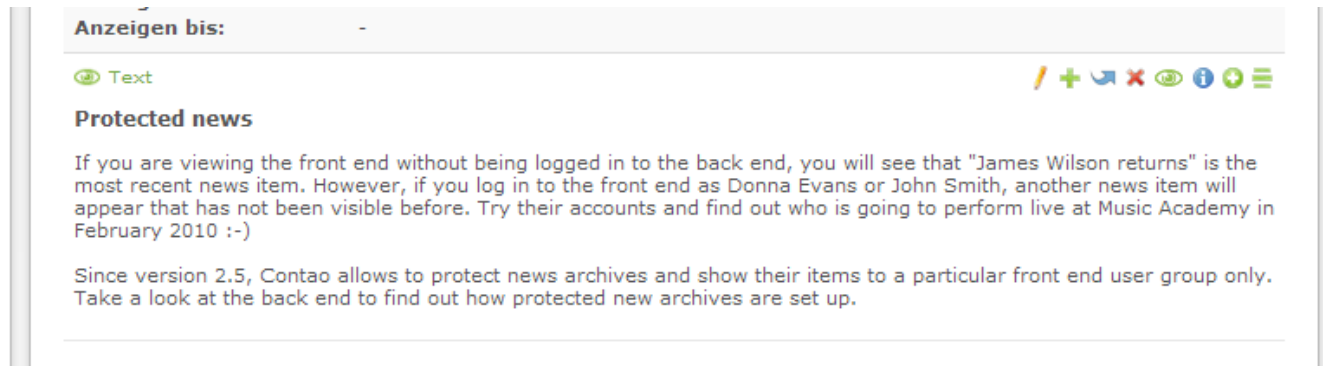


# DCA Syntax :: \$GLOBALS[...]['operations']

- \$GLOBALS['...']['operations']

Abschnitt für die Konfiguration der Aktionen pro Datensatz, bspw.

- Bearbeiten / Duplizieren / Verschieben / Löschen / ...



The screenshot shows a configuration interface for a news item. At the top, there is a field labeled "Anzeigen bis:" with a minus sign. Below this is a "Text" editor with a toolbar containing icons for undo, redo, bold, italic, link, unlink, and list. The main content area displays the text: "Protected news" followed by a paragraph: "If you are viewing the front end without being logged in to the back end, you will see that 'James Wilson returns' is the most recent news item. However, if you log in to the front end as Donna Evans or John Smith, another news item will appear that has not been visible before. Try their accounts and find out who is going to perform live at Music Academy in February 2010 :-)". Below this is another paragraph: "Since version 2.5, Contao allows to protect news archives and show their items to a particular front end user group only. Take a look at the back end to find out how protected new archives are set up."



# DCA Syntax :: \$GLOBALS[...]['operations']

```
1. <?php
2. $GLOBALS['TL_DCA']['tl_my_table']['list'] ...
3.
4.     'operations' => array(
5.         'edit' => array(
6.             'label' => &$GLOBALS['TL_LANG']['tl_my_table']['edit'],
7.             'href' => 'act=edit',
8.             'icon' => 'edit.gif',
9.         ),
10.        ...
11.    ),
12.
13. ...
```



# DCA Syntax :: \$GLOBALS[...]['fields']

- `$GLOBALS['...']['fields']`

## Abschnitt für die Konfiguration der einzelnen Felder

- Label und Beschreibung der Felder
- Verwendetes Eingabefeld (Text, Select, Checkbox, Radio, ...)
- Optionen für Select/Multi-Checkbox/Multi-Radio/...
- Detailsinstellungen, teilweise Eingabefeldspezifisch (eval Block)
  - CSS Klassen
- SQL Definition
- ...



# DCA Syntax :: \$GLOBALS[...]['fields']

```
1. <?php
2. $GLOBALS['TL_DCA']['tl_my_table'] ...
3.
4.     'fields' => array(
5.         'title' => array(
6.             'label'     => &$GLOBALS['TL_LANG']['tl_my_table']['title'],
7.             'exclude'   => true,
8.             'inputType' => 'text',
9.             'search'    => true,
10.            'eval'      => array('mandatory'=>true, 'maxlength'=>255,'decodeEntities'=>true),
11.            'sql'       => "varchar(255) NOT NULL default ''"
12.        ),
13.        ...
14.    )
15.
16. ...
```



# DCA Syntax :: \$GLOBALS[...]['palettes']

- `$GLOBALS['...']['palettes']`  
`$GLOBALS['...']['subpalettes']`

## Abschnitt für die Konfiguration der Anzeigemasken

- Angezeigte Felder
- Angezeigte Felder, wenn ein Feld X einen bestimmten Wert hat (siehe subpalettes)
- Reihenfolge der Felder
- Gruppierung der Felder (siehe Legenden)



# DCA Syntax :: \$GLOBALS[...]['palettes']

```
1. <?php
2. $GLOBALS['TL_DCA']['tl_my_table'] ...
3.
4.     'palettes' => array(
5.         '__selector__' => array('type', ...),
6.         'default'      => '{title_legend},title,alias,type',
7.     ),
8.
9. ...
```



## DCA Syntax :: \$GLOBALS[...]['metapalettes']

- `$GLOBALS['...']['metapalettes']`

### Achtung Eigenwerbung!

Die Extension [MetaPalettes](#) (ER2) bzw. [bit3/contao-meta-palettes](#) ermöglicht es die Paletten in Form von Arrays zu definieren.

Außerdem liefert sie einige Helper Methoden, zum manipulieren der String-basierten Paletten, siehe Wiki: [de.contaowiki.org/MetaPalettes](https://de.contaowiki.org/MetaPalettes)



# DCA Syntax :: \$GLOBALS[...][ 'metapalettes' ]

```
1. <?php
2. $GLOBALS['TL_DCA']['tl_my_table'] ...
3.
4.     'palettes' => array(
5.         '__selector__' => array('type', ...),
6.     ),
7.
8.     'metapalettes' => array(
9.         'default' => array(
10.            'title' => array('title', 'alias', 'type'),
11.        ),
12.    ),
13.
14.    ...
15.
```





# DCA Dokumentation

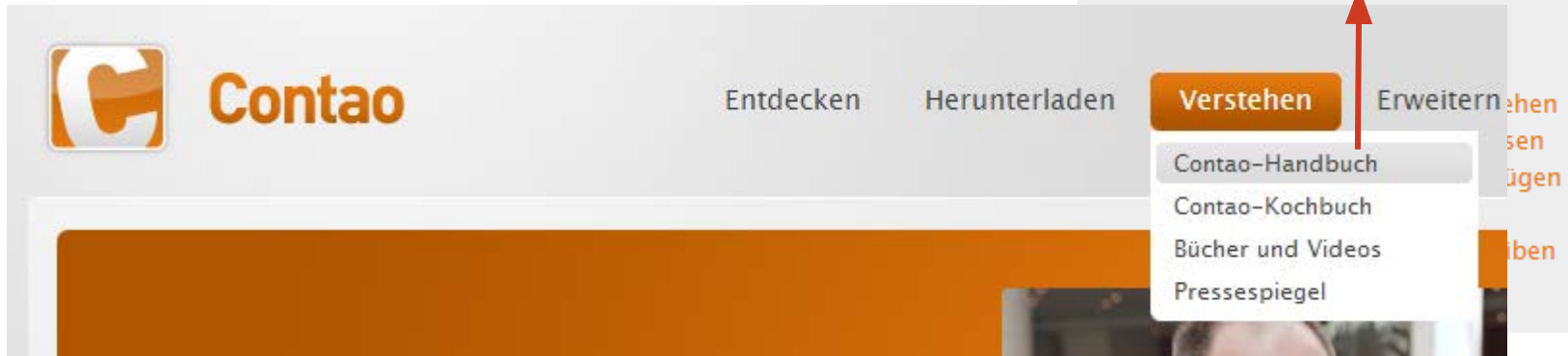
- Dokumentation / Referenz:
  - [contao.org](https://contao.org)
    - ↳ Verstehen
    - ↳ Contao Handbuch
    - ↳ Contao X.Y
    - ↳ [Data Container Arrays](#)

## V. Systemadministration

1. Benutzer und Gruppen
2. Erweiterungen
3. Systemwartung

## VI. Data Container Arrays

1. Referenz
2. Paletten
3. Callbacks



The screenshot shows the top navigation bar of the Contao website. The logo is on the left, followed by the text 'Contao'. To the right are the navigation items 'Entdecken', 'Herunterladen', 'Verstehen', and 'Erweitern'. The 'Verstehen' item is highlighted in orange, and a white dropdown menu is open below it, containing the following items: 'Contao-Handbuch', 'Contao-Kochbuch', 'Bücher und Videos', and 'Pressespiegel'. A red arrow points from the 'Data Container Arrays' section in the adjacent image to the 'Verstehen' button in this screenshot.




# DCA Dokumentation

- Ein (vollständiger) *undokumentierter* DCA:
  - [contao-community-alliance.github.io/dc-general-docs](https://contao-community-alliance.github.io/dc-general-docs)
    - ↳ Reference
    - ↳ Legacy DCA

*Dient uns aktuell als  
Gedächtnisstütze  
bei der Entwicklung  
des DC General.*

*Achtung: ein paar Felder sind  
DC General spezifisch!*



```
DcGeneral 1.0
Legacy DCA
This is a full legacy dca reference example with all known keys and example values.

$GLOBALS['TL_DCA']['tl_example'] = array
(
    // Config
    'config' => array
    (
        'label'           => &$GLOBALS['TL_LANG']['tl_example']['headline'],
        'dataContainer'   => 'General',
        'ptable'          => 'tl_parent',
        'dynamicPtable'   => true, // require 'ptable'=>''
        'ctable'          => array('tl_child1', 'tl_child2'),
        'validFileTypes'  => 'jpg,png,gif',
        'uploadScript'    => '',
        'closed'          => true,
        'notEditable'     => true,
        'notDeletable'    => true,
        'switchToEdit'    => true,
```



# database.sql (Contao <= 2.11)

```
1.  --
2.  -- Table `tl_my_table`
3.  --
4.  CREATE TABLE `tl_my_table` (
5.    `id` int(10) unsigned NOT NULL auto_increment,
6.    `tstamp` int(10) unsigned NOT NULL default '0',
7.    `title` varchar(255) NOT NULL default '',
8.    `alias` varbinary(128) NOT NULL default '',
9.    `keywords` text NULL,
10.   `email` varchar(128) NOT NULL default '',
11.   `salary` float(10,2) NOT NULL default '0.00',
12.   `published` char(1) NOT NULL default '',
13.   PRIMARY KEY (`id`),
14.   KEY `alias` (`alias`)
15. ) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```



# Klassen / Class-Loader



# Klassen

- Klassen können “irgendwo” im Modulverzeichnis liegen
  - Contao verwendet  
classes | elements | modules | models
  - Unsere empfehlung: PSR-0  
ab `system/modules/my_extension/src/`
- Hinweis: Klassen müssen in `autoload.php` registriert werden



# Contao Autoloader

```
1. <?php
2.
3. ClassLoader::addNamespaces(array(
4.     'My',
5. ));
```

- Möglichkeit Core Klassen zu Überschreiben
  - !!! Nur als **aller letzte** Möglichkeit !!!
  - !!! Nicht erlaubt in öffentlichen Erweiterungen !!!



# Contao Autoloader

```
1. <?php
2.
3. ClassLoader::addClasses(array(
4.     'My\Element' => 'system/modules/my_module/src/My/Element.php',
5.     'My\Module'  => 'system/modules/my_module/src/My/Module.php',
6. ));
```

- Enthält alle Klassen für das Module
- [Namespace\Klasse] => [Pfad zu der Datei]



# Contao Autoloader

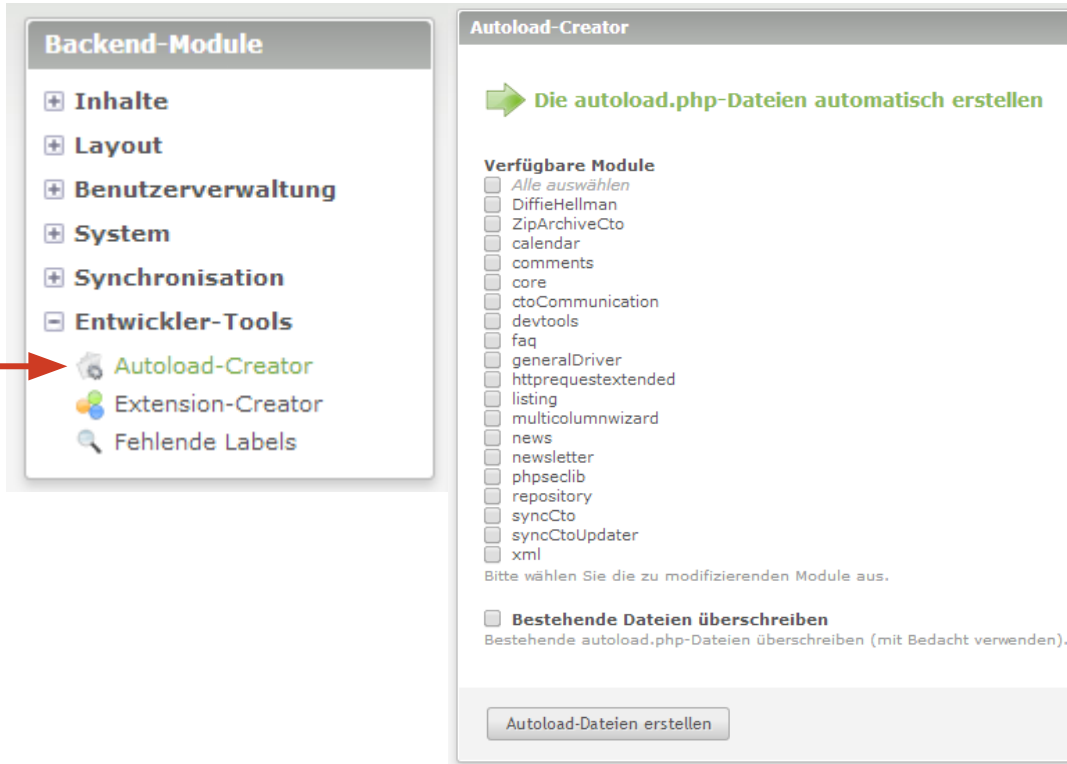
```
1. <?php
2.
3. TemplateLoader::addFiles(array(
4.     'ce_my_element' => 'system/modules/my_module/templates/elements', // Keine Dateiendung => php|html|xhtml
5.     'mod_my_module' => 'system/modules/my_module/templates/modules', // Nur Ordnerpfad
6. ));
```

- Enthält alle Templates für das Module
- [Template Name] => [Ordner Pfad]





# Developer Tools - Autoload Creator



The screenshot shows the Drupal Developer Tools interface. On the left, a sidebar titled "Backend-Module" lists various tool categories: Inhalte, Layout, Benutzerverwaltung, System, Synchronisation, and Entwickler-Tools. Under "Entwickler-Tools", the "Autoload-Creator" tool is highlighted with a red arrow. The main content area is titled "Autoload-Creator" and features a green arrow icon with the text "Die autoload.php-Dateien automatisch erstellen". Below this, there is a section "Verfügbare Module" with a list of modules and checkboxes: Alle auswählen, DiffieHellman, ZipArchiveCto, calendar, comments, core, ctoCommunication, devtools, faq, generalDriver, httprequestextended, listing, multicolumnwizard, news, newsletter, phpselib, repository, syncCto, syncCtoUpdater, and xml. A note below the list says "Bitte wählen Sie die zu modifizierenden Module aus." At the bottom, there is a checkbox for "Bestehende Dateien überschreiben" with the subtext "Bestehende autoload.php-Dateien überschreiben (mit Bedacht verwenden)." and a button labeled "Autoload-Dateien erstellen".

- Auswahl jeder Extension im "system/modules" Ordner
- Erstellen der autoload.php und autoload.ini
- Überschreiben von bestehenden Dateien

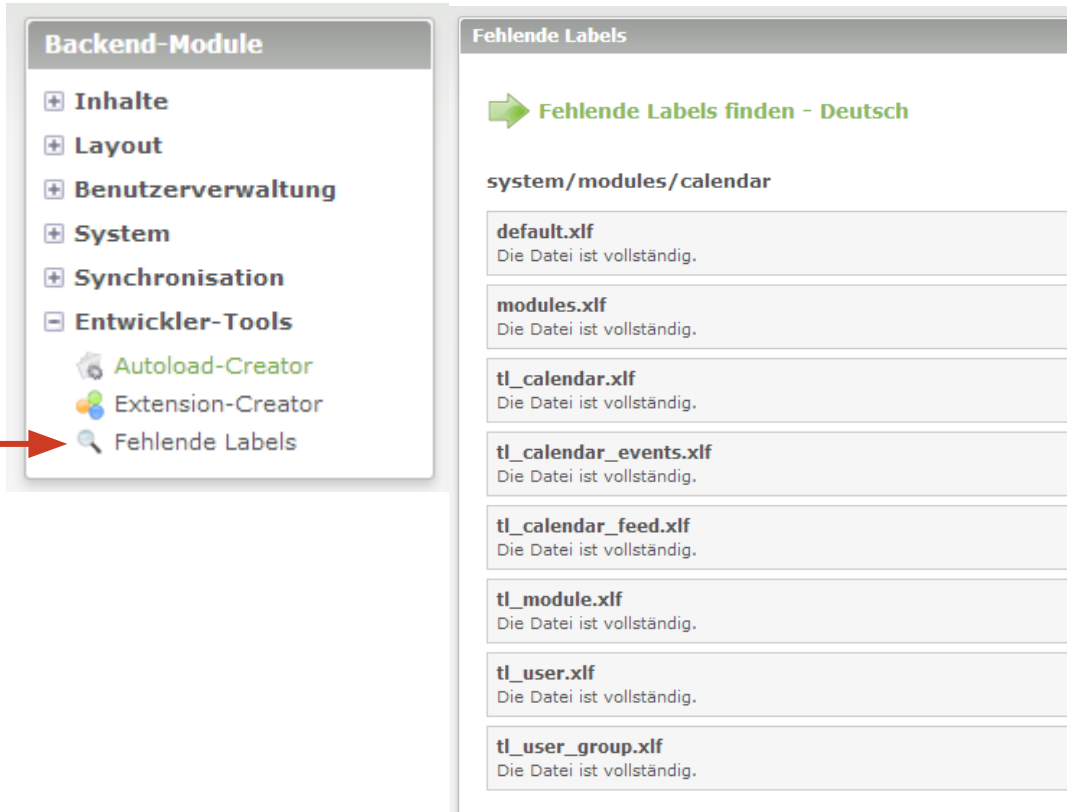


# config/autoload.ini

```
1.  ;;
2.  ; Configure what you want the autoload creator to register
3.  ;;
4.  register_namespaces = true
5.  register_classes    = true
6.  register_templates  = true
7.
8.  ;;
9.  ; Override the default configuration for certain sub directories
10. ;;
11. [vendor/*]
12. register_namespaces = false
13. register_classes    = false
14. register_templates  = false
```



# Developer Tools - Fehlende Labels



The screenshot shows the TYPO3 Developer Tools interface. On the left, a sidebar titled 'Backend-Module' lists various tools. The 'Entwickler-Tools' section is expanded, and 'Fehlende Labels' is selected, indicated by a red arrow. The main content area, titled 'Fehlende Labels', shows a green arrow icon and the text 'Fehlende Labels finden - Deutsch'. Below this, the path 'system/modules/calendar' is displayed. A list of XLIFF files is shown, each with a status of 'Die Datei ist vollständig.':

- default.xlf
- modules.xlf
- tl\_calendar.xlf
- tl\_calendar\_events.xlf
- tl\_calendar\_feed.xlf
- tl\_module.xlf
- tl\_user.xlf
- tl\_user\_group.xlf

- Auswahl der Sprache
  - Bis auf Englisch :)
- Anzeigen von fehlenden Labels
- Nur DCA Felder werden geprüft



# Framework



# Framework :: Database

```
1. <?php
2.
3. $database = \Database::getInstance(); // NICHT \Contao\Database !!!
4.
5. $result = $database->query('SELECT * FROM tl_table');
6.
7. $result = $database->prepare('SELECT * FROM tl_table WHERE id=?')->execute($id);
8.
9. while ($result->next()) {
10.     $row = $result->row();
11. }
```

[api.contao.org/classes/Contao.Database.html](https://api.contao.org/classes/Contao.Database.html)



# Framework :: Input

```
1. <?php
2.
3. $value = \Input::get('id'); // NICHT \Contao\Input !!!
4.
5. $value = \Input::post('id');
6.
7. $html = \Input::postHtml('text');
8.
9. $raw = \Input::postRaw('raw');
10.
11. $user = \Input::cookie('user');
```



# Framework :: Environment

```
1. <?php
2.
3. $scriptFilename = \Environment::get('scriptFilename'); // NICHT \Contao\Environment !!!
4. $scriptName     = \Environment::get('scriptName');
5. $phpSelf        = \Environment::get('phpSelf');
6. $documentRoot   = \Environment::get('documentRoot');
7. $requestUri     = \Environment::get('requestUri');
8. $ssl            = \Environment::get('ssl');
9. $url            = \Environment::get('url');
10. $base           = \Environment::get('base');
11. $host           = \Environment::get('host');
12. ...
```

[api.contao.org/classes/Contao.Environment.html](https://api.contao.org/classes/Contao.Environment.html) (siehe protected Methoden)



# Framework

## Weitere wichtige Klassen aus dem Framework

- **FrontendUser** → Authentifizierung von Mitgliedern im Frontend
- **BackendUser** → Authentifizierung von Benutzern im Backend
- **Session** → Zugriff auf Session Daten (werden in der DB langfristig gespeichert)
- **System / Controller / Frontend / Backend** → ein "Haufen" Hilfsfunktionen: Logging, URL-Generation, Template-Handling, ...
- **String / Image / Dbafs / File / Files / Folder**  
→ weitere Hilfsfunktionen





# Pro-Tipps



# DCA & Klassen in einer Datei

```
1.  $GLOBALS['TL_DCA']['tl_content'] = array
2.  (
3.      // Config
4.      'config' => array
5.      (
6.          'onload_callback' => array
7.          (
8.              array('tl_content',
9.                  'showJsLibraryHint')
10.         ),
11.         [...]
12.     )
13.
```

```
100. class tl_content extends Backend
101. {
102.     public function showJsLibraryHint($dc)
103.     {
104.         if ($_POST || Input::get('act') !=
105.             'edit')
106.             return;
107.     }
108.     [...]
109. }
110.
```





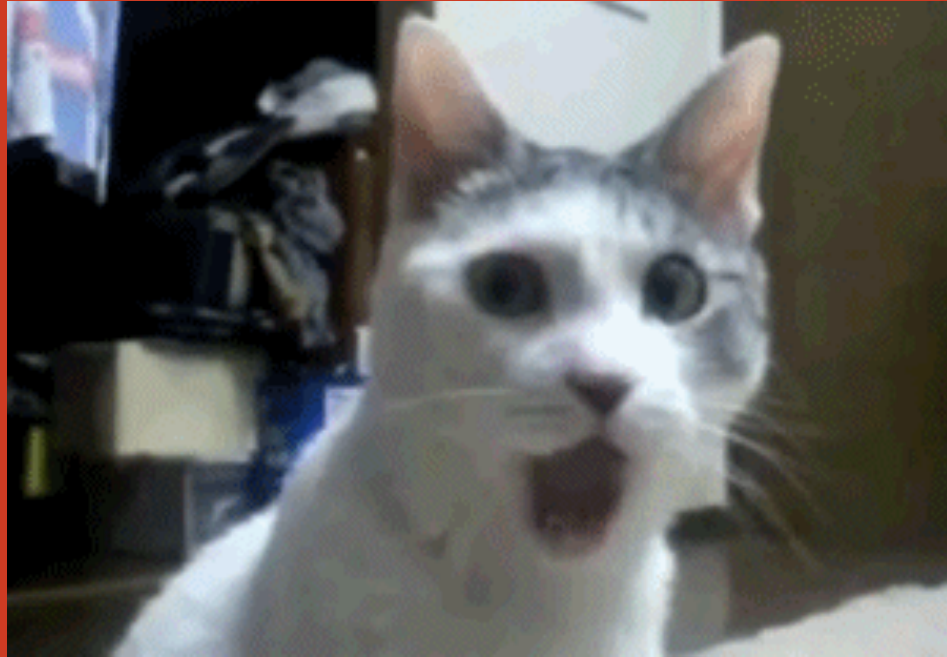
# !(DCA & Klassen in einer Datei)

- Probleme mit Erweitern von Klassen
- Probleme wenn DCA-Datei mehrfach geladen wird
  - Fatal error : Cannot redeclare class ...



**Ich brauche keine englischen Übersetzungen!**





## Doch brauchen wir....

- Fallback Sprache von Contao => Englisch
- Contao wird nicht nur in Deutschland benutzt
  - Siehe nächste Folie







# Klassenaufrufe in der config.php



# Klassenaufrufe in der config.php

```
1.  /**
2.   * Update something in the database
3.   */
4.  if (\Input::get('foo') == 'bar')
5.  {
6.      $database = \Database::getInstance();
7.      $database->prepare('...')->execute(\Input::get('id'));
8.  }
9.
```





# Probleme

- Weiße Seiten bei Fehlern in der config.php
- Kein Eintrag in der error.Log
- Kein Eintrag im Apache Log / FPM Log
- Schwer (/ Unmöglich) zu debuggen



# initializeSystem Hook

```
1. // config.php
2. $GLOBALS['TL_HOOKS']['initializeSystem'][] = array('MyClass', 'myInitializeSystem');
3.
4. // MyClass.php
5. class MyClass {
6.     public function hookInitializeSystem() {
7.         if (\Input::get('foo') == 'bar') {
8.             $database = \Database::getInstance();
9.             $database->prepare('...')->execute(\Input::get('id'));
10.        }
11.    }
12. }
```



# **Im Anschluss (gleicher Raum): Composer für Entwickler**



**Vielen Dank für das  
Zuschauen und Zuhören.**

**Diskussion**



