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

twitter.com/TristanLins

plus.google.com/+TristanLins

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

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

```php
<?php

ClassLoader::addNamespaces(array(
  'My',
));

ClassLoader::addClasses(array(
  'My\Element' => 'system/modules/my_module/src/My/Element.php',
  'My\Module'  => 'system/modules/my_module/src/My/Module.php',
));

TemplateLoader::addFiles(array(
  'ce_my_element' => 'system/modules/my_module/templates/elements',
  'mod_my_module' => 'system/modules/my_module/templates/modules',
));
```

config/autoload.ini

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

config/config.php

```php
<?php
/** Back end modules */
$GLOBALS['BE_MOD']['my_category']['my_menu_item'] = array('tables' => array('tl_my_table'));
/** Front end modules */
$GLOBALS['FE_MOD']['my_category']['my_frontend_module'] = 'My\Module';
/** Content elements */
$GLOBALS['TL_CTE']['my_category']['my_content_element'] = 'My\Element';
/** Back end form fields */
$GLOBALS['BE_FFL']['my_field'] = 'My\BackEndFormField';
/** Front end form fields */
$GLOBALS['TL_FFL']['my_field'] = 'My\FrontendFormField';
/** Hooks */
$GLOBALS['TL_HOOKS']['loadLanguageFile']['my_extension'] = array('My\Hooks', 'loadLanguageFile');
/** Models */
$GLOBALS['TL_MODELS']['tl_my_table'] = 'My\TableModel';
```

config/config.php

```php
<?php

// Version der eigenen Extension registrieren
define('MY_EXTENSION_VERSION', '1.2.3');

// Eigene Konfigurationsvariablen mit Standardwerten befüllen
// (kann in der localconfig.php überschrieben werden)
$GLOBALS['TL_CONFIG']['myConfigEntry'] = 'my default value';

// Eigene Register-Arrays definieren
// KEIN ... => array() ... verwenden!
if (!isset($GLOBALS['MY_EXTENSION_REGISTER'])) {
    $GLOBALS['MY_EXTENSION_REGISTER'] = array();
}
$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>/*

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xliff version="1.1">
  <file datatype="php"
        original="system/modules/my_extension/
                  languages/en/modules.php"
        source-language="en">
    <body>
      <trans-unit id="MOD.my_extension">
        <source>My extension</source>
      </trans-unit>
    </body>
  </file>
</xliff>
```

```php
<?php

$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

```php
<?php

$GLOBALS['TL_DCA']['tl_my_table'] = array(

        'config' => array( ... ),

        'list' => array( ... ),

        'palettes' => array( ... ),
        'subpalettes' => array( ... ),

        'fields' => array( ... ),

);
```

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 / löschbar / sortierbar / ...)
  - Callbacks beim laden / speichern / löschen /... eines Datensatzes
  - SQL Indizes
  - ...

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

```php
<?php
$GLOBALS['TL_DCA']['tl_my_table'] ...

        'config' => array(
                'label'         => $GLOBALS['TL_LANG']['tl_my_table']['containerHeadline'],
                'dataContainer' => 'Table',
                'sql' => array(
                        'keys' => array(
                                'id'    => 'primary',
                                'alias' => 'index',
                                'email' => 'unique',
                        )
                )
        ),

...
```

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)
  - ...

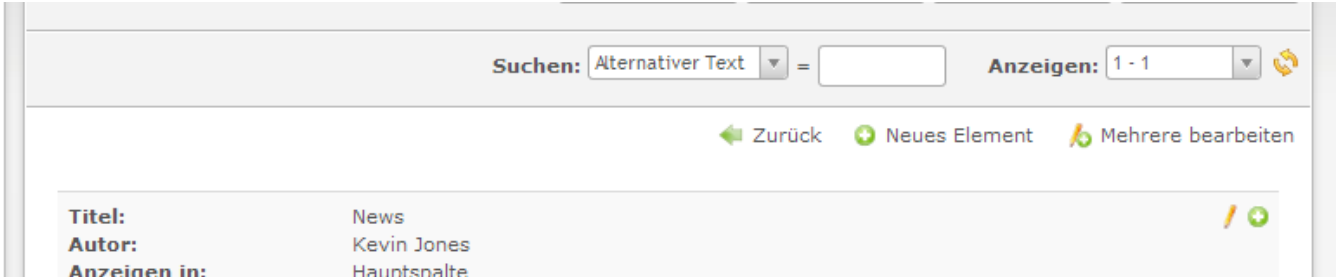
DCA Syntax :: $GLOBALS[...]['list']

```php
<?php
$GLOBALS['TL_DCA']['tl_my_table'] ...

        'list' => array(
                'sorting' => array(
                        'mode' => 5,
                ),
                'label' => array(
                        'fields'         => array('title'),
                        'format'         => '%s',
                ),
                'global_operations' => array( ... ),
                'operations'        => array( ... )
        ),

...
```

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



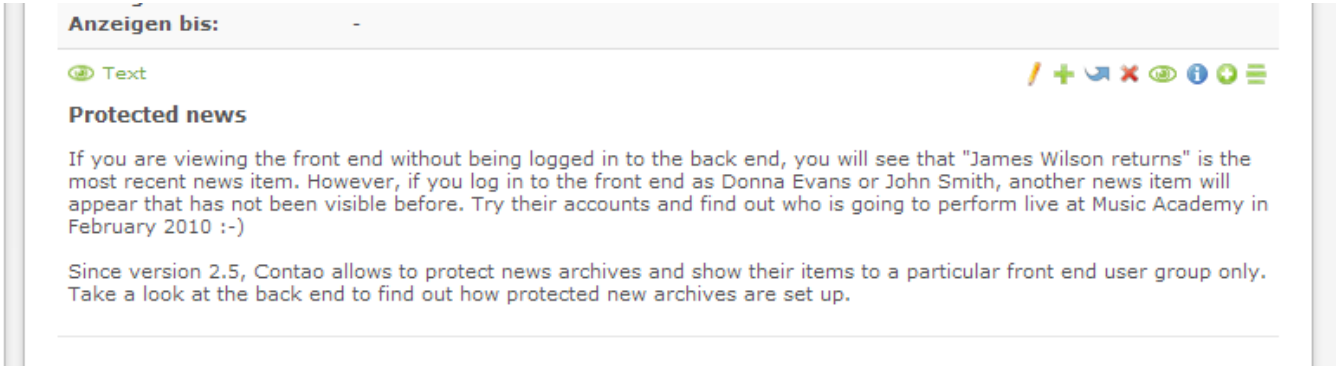
DCA Syntax :: $GLOBALS[...]['global_operations']

```php
<?php
$GLOBALS['TL_DCA']['tl_my_table']['list'] ...

        'global_operations' => array(
                'all' => array(
                        'label'      => &$GLOBALS['TL_LANG']['MSC']['all'],
                        'href'       => 'act=select',
                        'class'      => 'header_edit_all',
                        'attributes' => 'onclick="Backend.getScrollOffset()" accesskey="e"'
                ),
                ...
        ),

...
```

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

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

  Abschnitt für die Konfiguration der Aktionen pro Datensatz, bspw.
  - Bearbeiten / Duplizieren / Verschieben / Löschen / ...



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/...
  - Detaileinstellungen, teilweise Eingabefeldspezifisch (eval Block)
    - CSS Klassen
  - SQL Definition
  - ...

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

```php
<?php
$GLOBALS['TL_DCA']['tl_my_table'] ...

        'fields' => array(
                'title' => array(
                        'label'     => &$GLOBALS['TL_LANG']['tl_my_table']['title'],
                        'exclude'   => true,
                        'inputType' => 'text',
                        'search'    => true,
                        'eval'      => array('mandatory'=>true, 'maxlength'=>255,'decodeEntities'=>true),
                        'sql'       => "varchar(255) NOT NULL default ''"
                ),
                ...
        )

...
```

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']

```php
<?php
$GLOBALS['TL_DCA']['tl_my_table'] ...

        'palettes' => array(
                '__selector__' => array('type', ...),
                'default'      => '{title_legend},title,alias,type',
        ),

...
```

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

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

```php
<?php
$GLOBALS['TL_DCA']['tl_my_table'] ...

        'palettes' => array(
                '__selector__' => array('type', ...),
        ),

        'metapalettes' => array(
                'default' => array(
                        'title' => array('title', 'alias', 'type'),
                ),
        ),

...

```

DCA Dokumentation

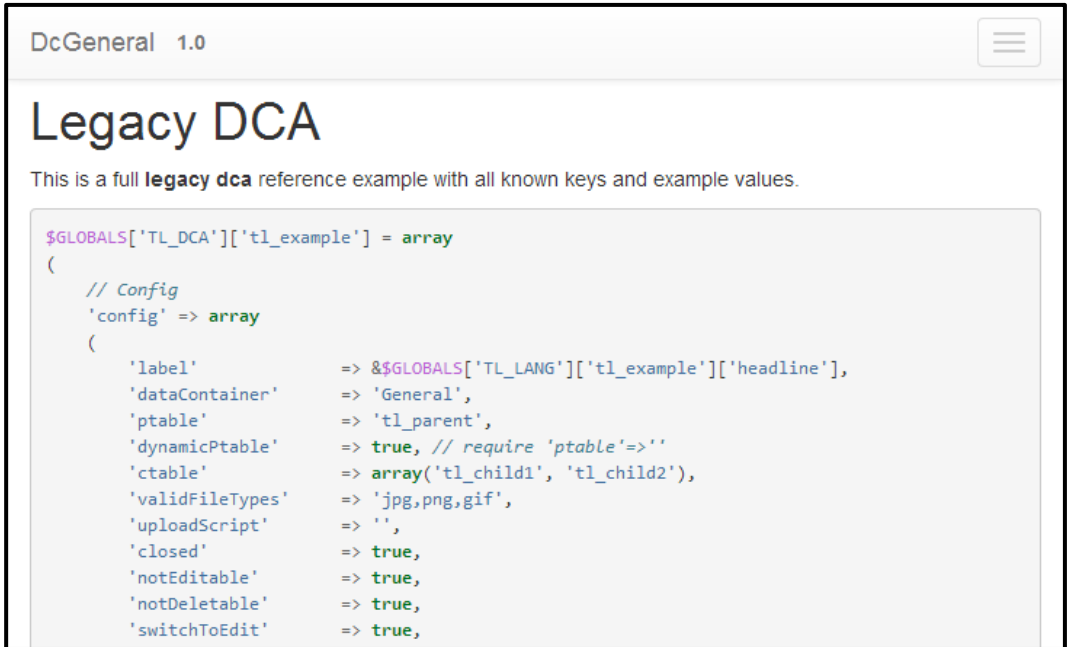
- Dokumentation / Referenz:
  - contao.org
    - ↳ Verstehen
    - ↳ Contao Handbuch
    - ↳ Contao X.Y
    - ↳ Data Container Arrays

DCA Dokumentation

- Ein (vollständiger) *undokumentierter* DCA:
  - 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!



database.sql (Contao <= 2.11)

```sql
--
-- Table `tl_my_table`
--
CREATE TABLE `tl_my_table` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `tstamp` int(10) unsigned NOT NULL default '0',
  `title` varchar(255) NOT NULL default '',
  `alias` varbinary(128) NOT NULL default '',
  `keywords` text NULL,
  `email` varchar(128) NOT NULL default '',
  `salary` float(10.2) NOT NULL default '0.00',
  `published` char(1) NOT NULL default '',
  PRIMARY KEY  (`id`),
  KEY `alias` (`alias`)
) 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

```php
<?php

ClassLoader::addNamespaces(array(
  'My',
));
```

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

Contao Camp 2012 Vortrag zum Thema Namespaces in Contao auf YouTube

Contao Autoloader

```php
<?php

ClassLoader::addClasses(array(
  'My\Element' => 'system/modules/my_module/src/My/Element.php',
  'My\Module'  => 'system/modules/my_module/src/My/Module.php',
));
```

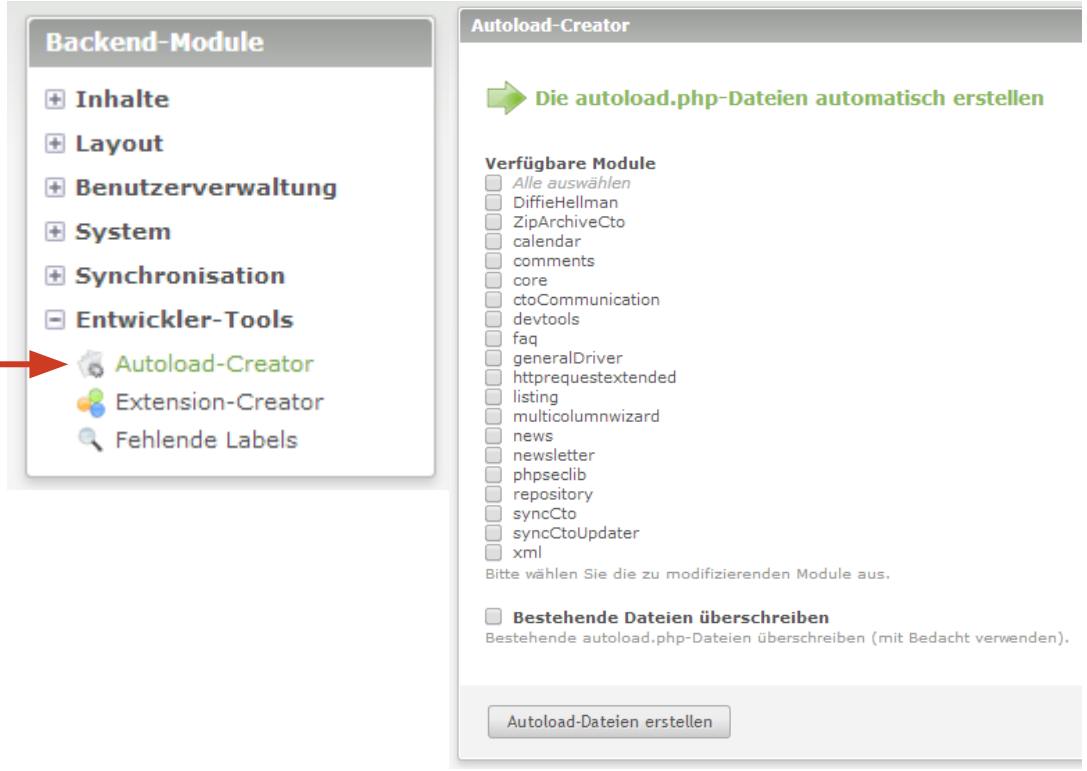
- Enthält alle Klassen für das Module
- [Namespace\Klasse] => [Pfad zu der Datei]

Contao Autoloader

```php
<?php

TemplateLoader::addFiles(array(
  'ce_my_element' => 'system/modules/my_module/templates/elements',  // Keine Dateiendung => php|html5|xhtml
  'mod_my_module' => 'system/modules/my_module/templates/modules',   // Nur Ordnerpfad
));
```

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

Developer Tools - Autoload Creator

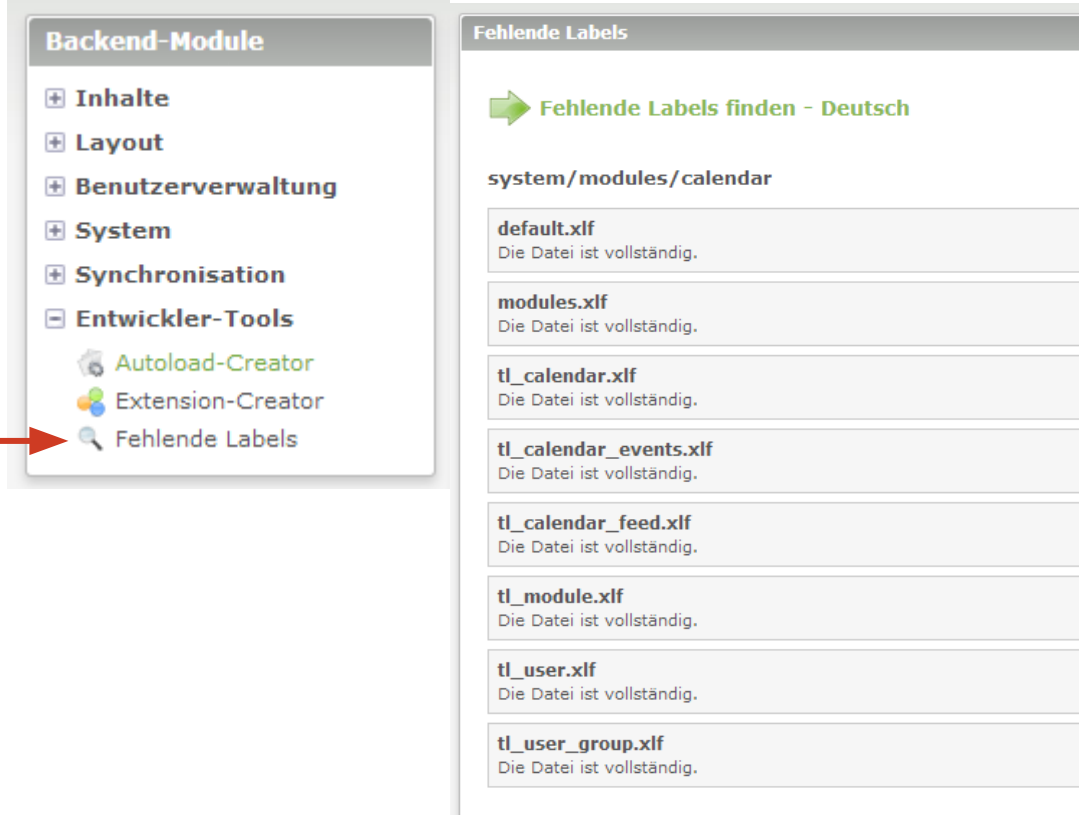


- Auswahl jeder Extension im “system/modules” Ordner
- Erstellen der autoload.php und autoload.ini
- Überschreiben von bestehenden Dateien

config/autoload.ini

```
;;
; Configure what you want the autoload creator to register
;;
register_namespaces = true
register_classes    = true
register_templates  = true

;;
; Override the default configuration for certain sub directories
;;
[vendor/*]
register_namespaces = false
register_classes    = false
register_templates  = false
```

Developer Tools - Fehlende Labels



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

Framework

Framework :: Database

```php
<?php

$database = \Database::getInstance(); // NICHT \Contao\Database !!!

$result = $database->query('SELECT * FROM tl_table');

$result = $database->prepare('SELECT * FROM tl_table WHERE id=?')->execute($id);

while ($result->next()) {
        $row = $result->row();
}
```

api.contao.org/classes/Contao.Database.html

Framework :: Input

```php
<?php

$value = \Input::get('id'); // NICHT \Contao\Input !!!

$value = \Input::post('id');

$html = \Input::postHtml('text');

$raw = \Input::postRaw('raw');

$user = \Input::cookie('user');
```

api.contao.org/classes/Contao.Input.html

Framework :: Environment

```php
<?php

$scriptFilename = \Environment::get('scriptFilename'); // NICHT \Contao\Environment !!!
$scriptName     = \Environment::get('scriptName');
$phpSelf        = \Environment::get('phpSelf');
$documentRoot   = \Environment::get('documentRoot');
$requestUri     = \Environment::get('requestUri');
$ssl            = \Environment::get('ssl');
$url            = \Environment::get('url');
$base           = \Environment::get('base');
$host           = \Environment::get('host');
...
```

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

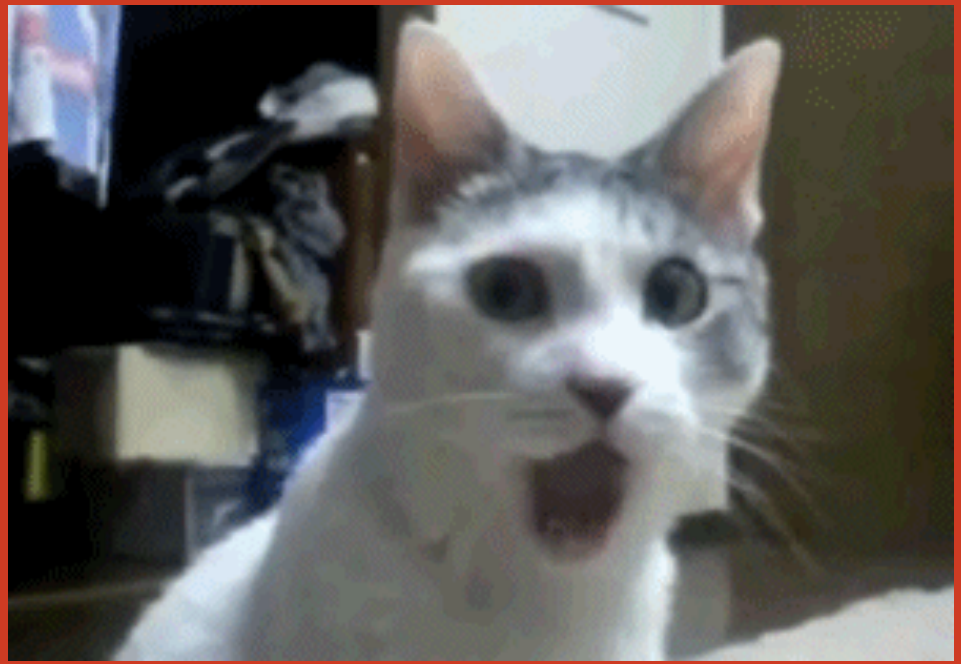
```php
$GLOBALS['TL_DCA']['tl_content'] = array
(
        // Config
        'config' => array
        (
                'onload_callback'             => array
                (
                        array('tl_content', 'showJsLibraryHint')
                ),
        )
        [...]
)
```

```php
class tl_content extends Backend
{
        public function showJsLibraryHint($dc)
        {
                if ($_POST || Input::get('act') != 'edit')
                {
                        return;
                }
                [...]
        }
}
```

!(DCA & Klassen in einer Datei)

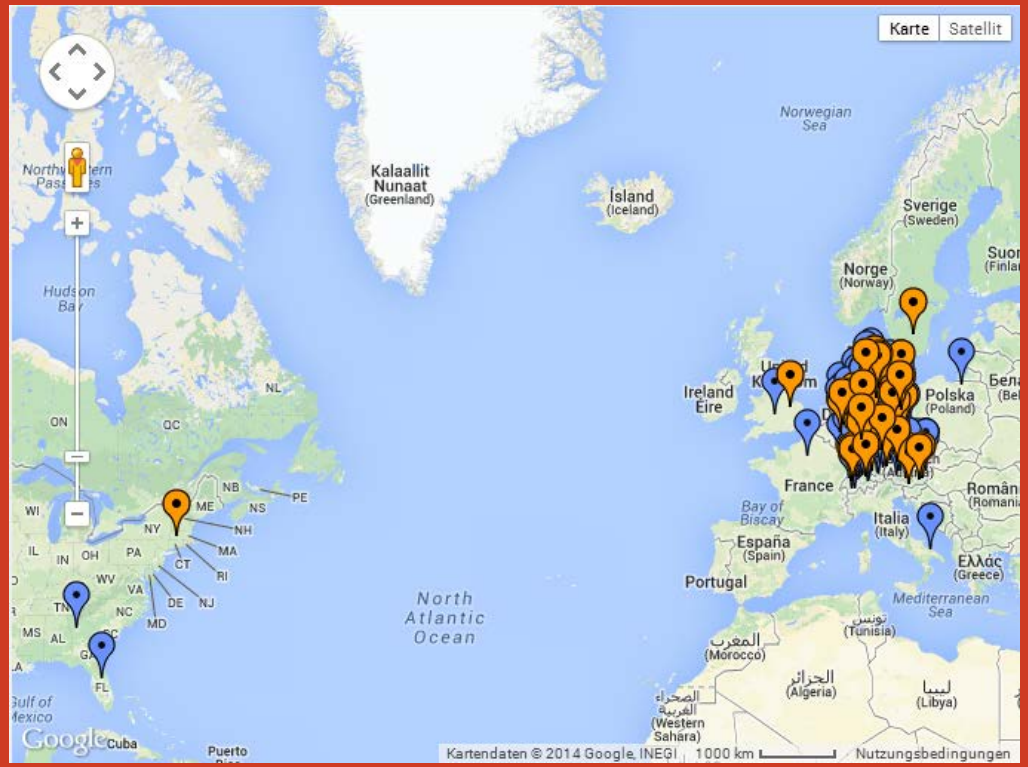
- Probleme mit Erweitern von Klassen
- Probleme wenn DCA-Datei mehrfach geladen wird
  - Fatal error : Cannot redeclare class ...

Ich brauche keine englischen Übersetzungen!



Doch brauchen wir....

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


Karte
Satellit
Kalaallit Nunaat (Greenland)
Island (Iceland)
Norwegian Sea
Sverige (Sweden)
Norge (Norway)
Polska (Poland)
Ireland Eire
France
Bay of Biscay
España (Spain)
Portugal
Italia (Italy)
Ελλάς (Greece)
Mediterranean Sea
North Atlantic Ocean
Hudson Bay
Gulf of Mexico
Cuba
Puerto
Kartendaten © 2014 Google, INEGI
1000 km
Nutzungsbedingungen


Takahiro Kambe
Sapporo
札幌
Niigata
新潟
Sendai
仙台
日本
(Japan)
Okayama
岡山
Osaka
大阪
Nagoya
名古屋
Tokyo
東京
Fukuoka
福岡
Hiroshima
広島
Kumamoto
熊本
Kagoshima
鹿児島

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.
```


livememe.com

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

```php
// config.php
$GLOBALS['TL_HOOKS']['initializeSystem'][] = array('MyClass', 'myInitializeSystem');

// MyClass.php
class MyClass {
        public function hookInitializeSystem() {
                if (\Input::get(‘foo’) == ‘bar’) {
                        $database = \Database::getInstance();
                        $database->prepare('...')->execute(\Input::get('id'));
                }
        }
}
```

contao.org/de/manual/3.2/customizing-contao.html

Im Anschluss (gleicher Raum): Composer für Entwickler

Vielen Dank für das Zuschauen und Zuhören.

Diskussion