Contao

Contao 3

Entwickler-Workshop 2012

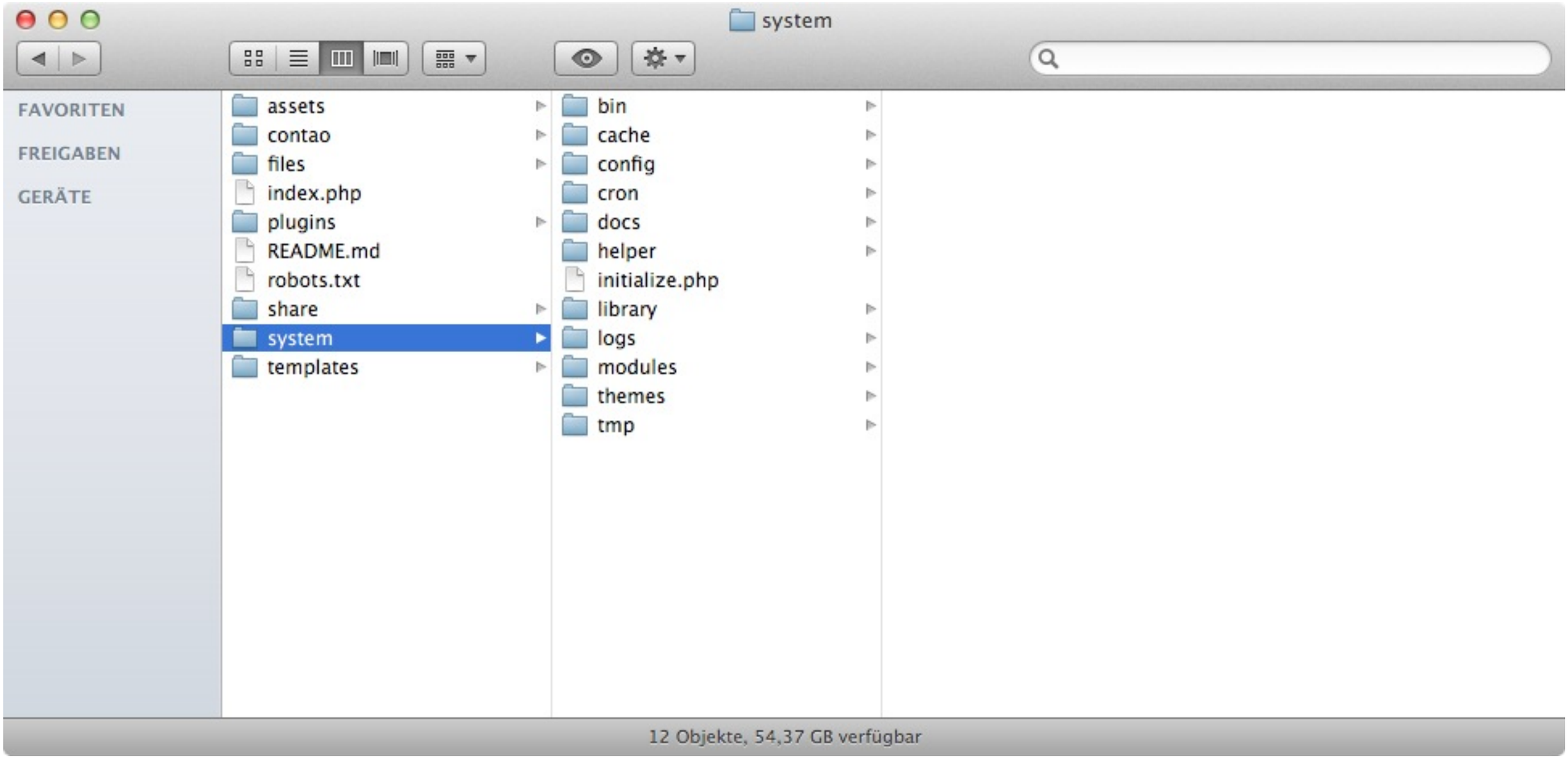
Übersicht

- Teil 1: Dateistruktur
- Teil 2: Class-Autoloading
- Teil 3: Models
- Teil 4: API-Änderungen
- Teil 5: Sonstiges

Teil 1

Die neue Dateistruktur in Contao 3

- Der System-Ordner

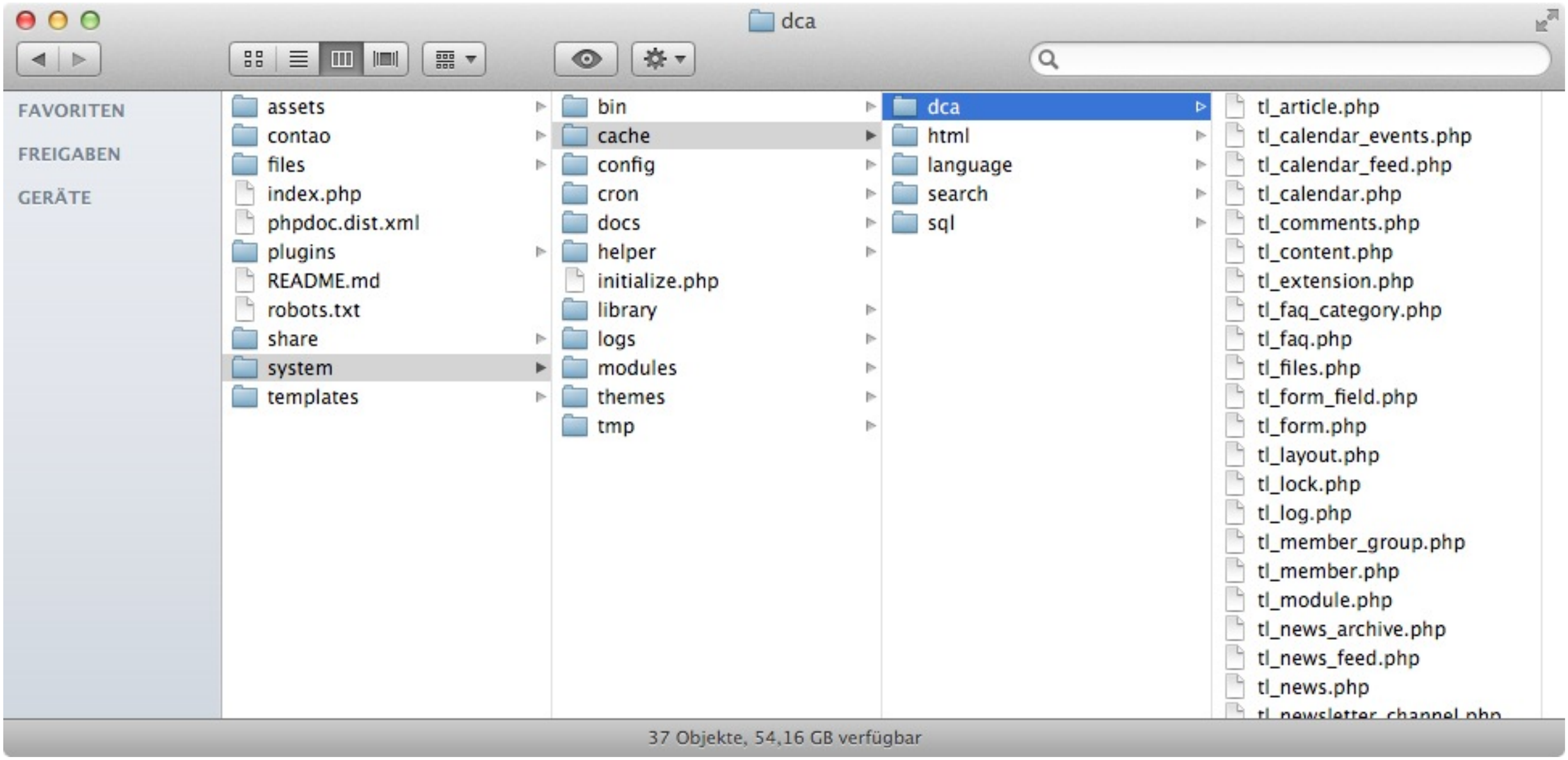


- system/bin/minify
  - Komprimierung der .js- und .css-Dateien
  - Muss auf der Kommandozeile aufgerufen werden
    - → ./system/bin/minify mootools
- Wie wird komprimiert?
  - .css-Dateien werden mit dem YUI-Compressor komprimiert
  - .js-Dateien werden mit UglifyJS oder dem YUI-Compressor komprimiert
- Was muss installiert werden?
  - Der YUI-Compressor wird mitgeliefert
  - Für UglifyJS muss node.js und npm installiert sein
- Warum brauche ich das?
  - Falls Plugins angepasst werden müssen, können trotzdem die komprimierten Dateien verwendet werden (Page Speed)

core — bash — 122×45

```
hqs:core leofeyer$ ./system/bin/minify

  Minify scripts using YUI compressor or UglifyJS (requires node.js)
  Usage: ./system/bin/minify <command> [<file>]

  <command>
  all             Execute all the functions

  Contao
  assets          Minify the Contao assets (assets/contao)
  contao          Minify the Contao back end script (system/modules/core/html/contao.js)
  theme           Minify the default back end theme (system/themes/default)
  typolinks       Minify the TYPOlinks plugin (plugins/tinyMCE/plugins/typolinks)

  jQuery
  jquery          Minify the jQuery core (plugins/jquery/core|ui)
  colorbox        Minify the colorbox plugin (plugins/jQuery/colorbox)
  mediaelement    Minify the mediaelement plugin (plugins/jQuery|mootools/mediaelement)

  MooTools
  mootools        Minify the MooTools core (plugins/mootools/core)
  chosen          Minify the chosen plugin (plugins/mootools/chosen)
  colorpicker     Minify the colorpicker plugin (plugins/mootools/colorpicker)
  datepicker      Minify the datepicker plugin (plugins/mootools/datepicker)
  mediabox        Minify the mediabox plugin (plugins/mootools/mediabox)
  mediaelement    Minify the mediaelement plugin (plugins/jQuery|mootools/mediaelement)
  mootao          Minify the Contao MooTools enhancements (plugins/mootools/mootao)
  simplemodal     Minify the SimpleModal plugin (plugins/mootools/simplemodal)
  slimbox         Minify the slimbox plugin (plugins/mootools/slimbox)
  stylect         Minify the stylect plugin (plugins/mootools/stylect)
  tablesort       Minify the tablesort plugin (plugins/mootools/tablesort)

  Other
  codemirror      Minify the CodeMirror plugin (plugins/codeMirror)
  highlighter     Minify the SyntaxHighlighter plugin (plugins/highlighter)

hqs:core leofeyer$
```

- Der interne Cache (system/cache)

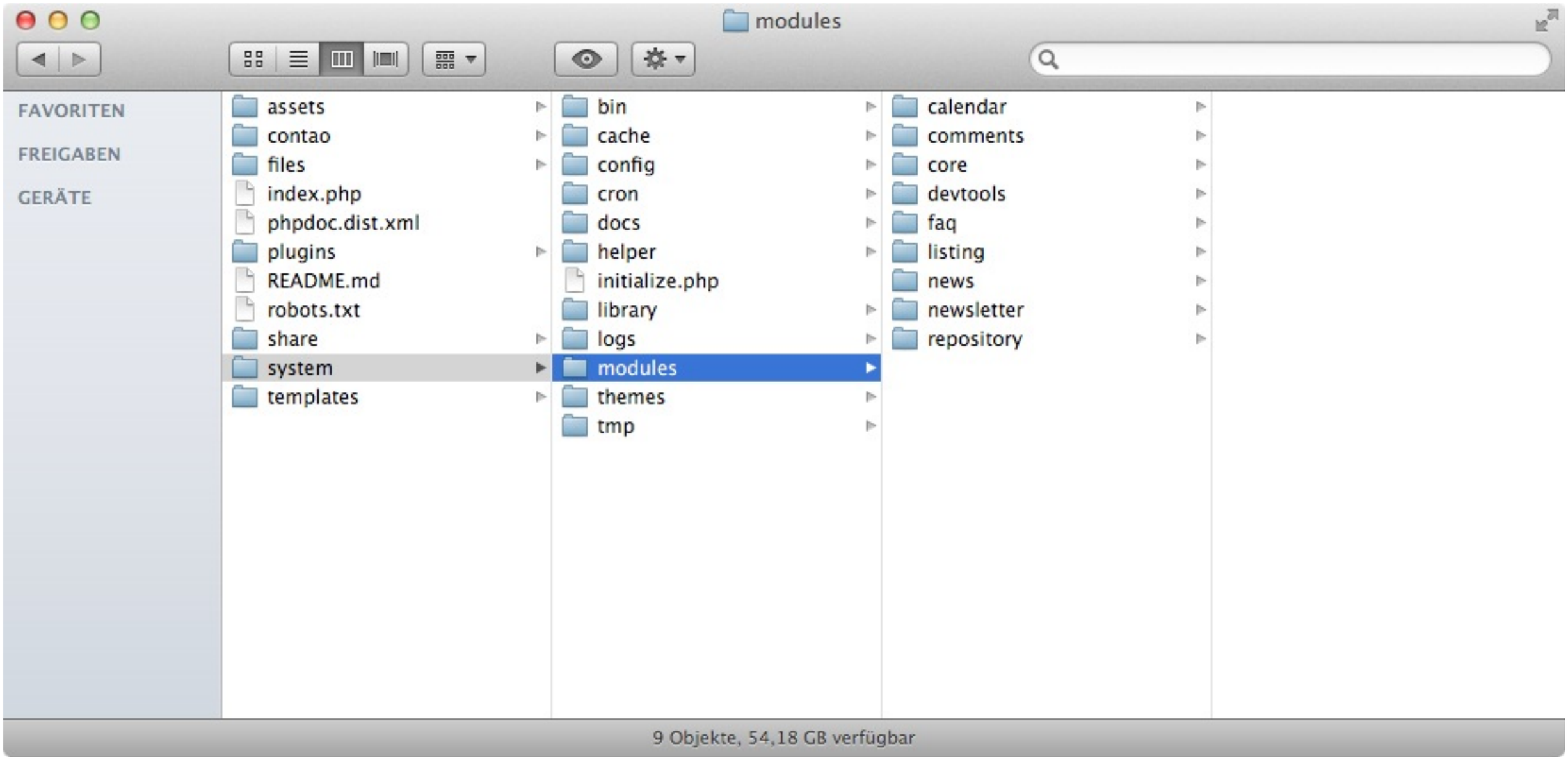


- Was wird gecacht?
  - system/dca Zusammenfassung der einzelnen Modul-DCA-Dateien
  - system/html Fertig gerenderte Frontend-Seiten
  - system/language Zusammenfassung der einzelnen Modul-Sprachdateien
  - system/search Suchanfragen
  - system/sql Die SQL-Anweisungen und Relationen aus den DCA-Dateien

- Warum wird gecacht?
  - Weniger Festplattenzugriffe
  - Weniger RAM-Verbrauch beim Auslesen der SQL-Anweisungen

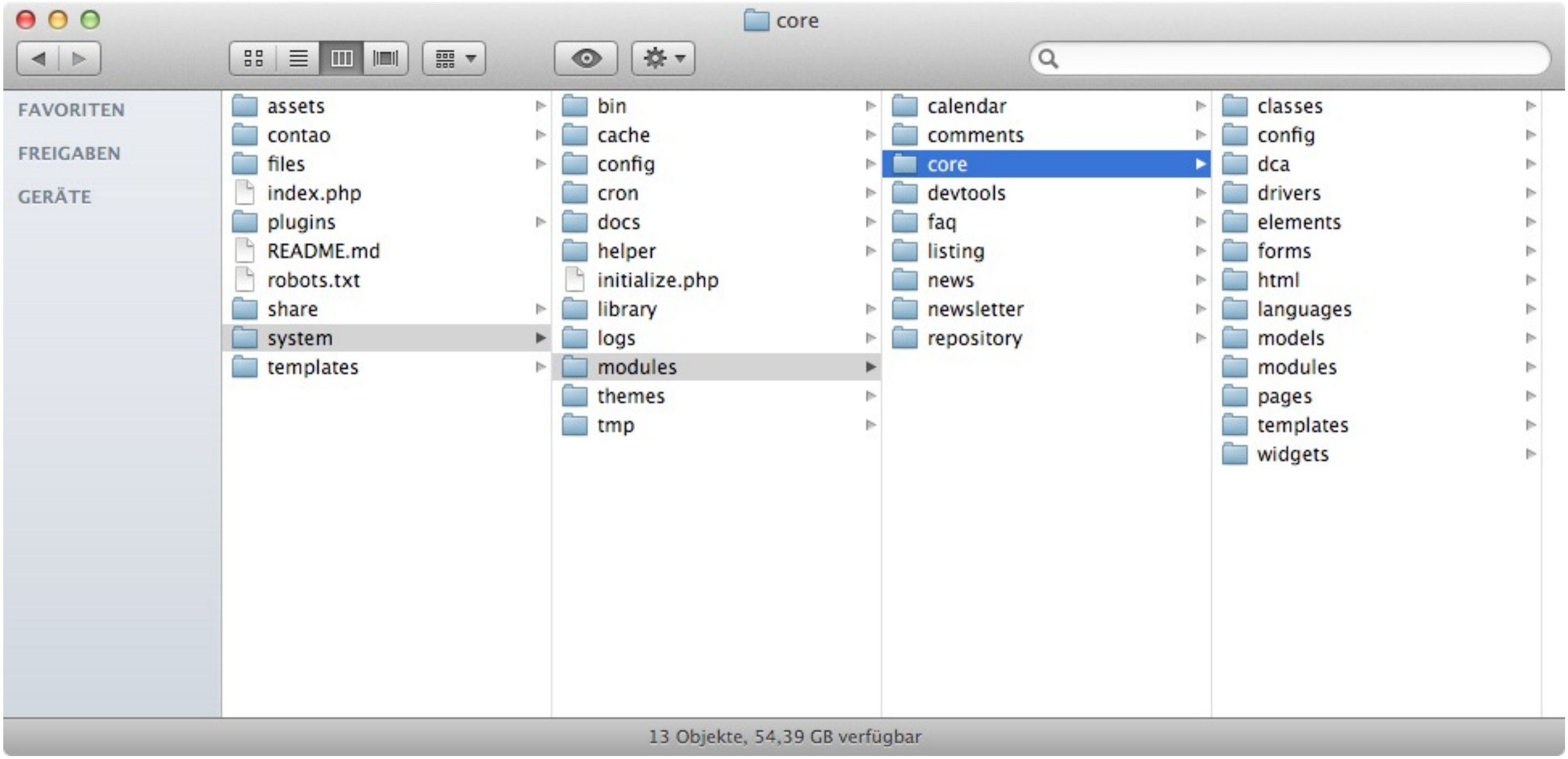
- Leeren des Cache
  - Über die Systemwartung
  - Über den Automator (Command Scheduler)

- Die Core-Module



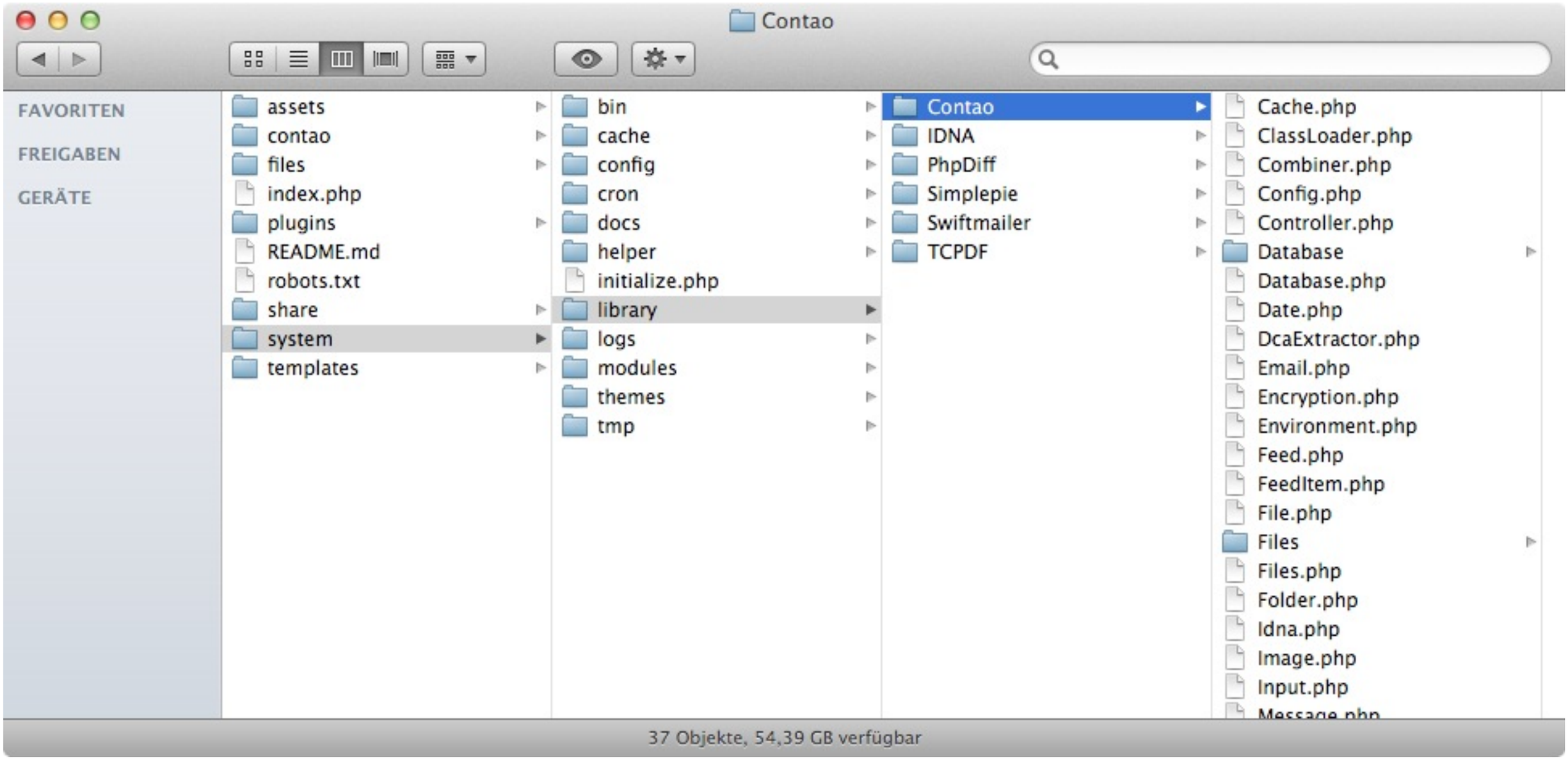
- Deutlich aufgeräumter als in Version 2
  - „backend“ und „frontend“ wurden als „core“ zusammengefasst
  - „registration“, „rss_reader“ und „tpl_editor“ wurden in „core“ integriert
  - „rep_base“ und „rep_client“ wurden als „repository“ zusammengefasst
- Das Entwickler-Modul ist wieder im Core
  - Das „development“-Modul wird wieder mit dem Core ausgeliefert
  - Um Missverständnisse zu vermeiden, wurde es in „devtools“ umbenannt
  - Das Modul wird in Version 3 auch für Anwender relevant

- Das Modul „core“



- Beliebige Unterordner möglich
  - Bessere Strukturierung der PHP-Dateien
  - Der Mapper-Class-Loader weiß, wo die Dateien liegen
  - Beliebige Unterordner (nicht verschachtelt) möglich
- Aufteilung der PHP-Klassen im Core
  - classes — Alles, was nicht in die anderen Kategorien passt
  - drivers — DC_-Dateien (DC_Table, DC_Folder etc.)
  - elements — Inhaltselemente
  - forms — Formular-Felder (Frontend)
  - models — Model-Klassen
  - modules — Frontend-Module
  - pages — Seiten-Klassen (PageRegular, PageRoot etc.)
  - widgets — Formular-Felder (Backend)

- Die Contao-Library



- Zentraler Library-Ordner
  - Im Ordner „system/library“ befinden sich alle PHP-Libraries
  - Neben dem Contao-Framework liegen dort z.B. auch Swiftmailer und TCPDF
  - Im Ordner „plugins“ liegen nur noch JavaScript-Plugins
- Bekannte Strukturen
  - Die Struktur des Contao-Frameworks hat sich nicht geändert
  - Alle Änderungen sind optional und weitestgehend rückwärtskompatibel

Teil 2

Class-Autoloading in Contao 3

- Mapper-Class-Loader
  - Bisher: Durchlaufen der Verzeichnisse auf der Suche nach einer Klasse
  - Neu: Die Klassen werden registriert und in einem Mapper-Array gespeichert
- Vorteile
  - Schnelleres Auffinden der Klassen
  - Deutliche weniger Festplattenzugriffe
  - Beliebige Unterordner zur Strukturierung möglich
  - Auch Templates laden dank Mapper schneller
- Warum überhaupt?
  - Keine einheitlichen Standards für die Benennung von Klassen
  - Umstellung auf PSR-0 aus Gründen der Rückwärtskompatibilität nicht möglich
  - Grosser Gewinn: Alle (!) Core-Klassen lassen sich überschreiben

- config/autoload.php
  - In der autoload.php-Datei werden die Klassen und Templates registriert
  - Die Erstellung der Datei erfolgt automatisch über den Autoload-Creator (devtools)
- Idealfall
  - Da die API in Contao 3 weitestgehend rückwärtskompatibel ist, reicht es oft schon aus, die autoload.php-Datei erstellen zu lassen, damit eine Contao 2-Erweiterung auch unter Contao 3 läuft
  - Getestet u.a. mit der Subcolumns-Erweiterung
- Portierung von Erweiterungen
  - Erweiterungen, die nicht mehr gepflegt werden, aber z.B. unter Contao 2.11 laufen, können mit dem Autoload-Creator portiert werden
  - Ausnahme: komplexe Erweiterungen mit dynamisch erstellten Klassen

Autoload-Creator – Contao | Welcome to Music Academy

localhost/contao/core/contao/main.php?do=autoload

Contao Open Source CMS 3.0.beta1

Benutzer k.jones :: Frontend-Vorschau :: Startseite :: Abmelden

Backend-Module

- **Inhalte**
  - Artikel
  - Nachrichten
  - Events
  - FAQ
  - Newsletter
  - Formulargenerator
  - Kommentare
- **Layout**
  - Themes
  - Seitenstruktur
  - Templates
- **Benutzerverwaltung**
  - Mitglieder
  - Mitgliedergruppen
  - Benutzer
  - Benutzergruppen
- **System**
  - Dateiverwaltung
  - System-Log
  - Einstellungen
  - Systemwartung
  - Wiederherstellen
  - Erweiterungskatalog
  - Erweiterungsverwaltung
- **Entwickler-Tools**
  - Extension-Creator

Autoload-Creator

Zurück

Die autoload.php-Dateien automatisch erstellen

Verfügbare Module

- [ ] *Alle auswählen*
- [ ] calendar
- [ ] comments
- [ ] core
- [ ] devtools
- [x] faq
- [ ] listing
- [ ] news
- [ ] newsletter
- [ ] repository

Optionen

- [x] Bestehende Autoload-Dateien überschreiben
- [ ] Die IDE-Kompatibilitätsdatei aktualisieren

Hier können Sie die *config/autoload.php*-Dateien, die für Contao 3 gebraucht werden und die Klassen und Templates zum Autoloader hinzufügen, automatisch erstellen. Falls in einer PHP-Klasse der *namespace*-Befehl gefunden wird, wird der Namensraum ebenfalls hinzugefügt.

Weiter

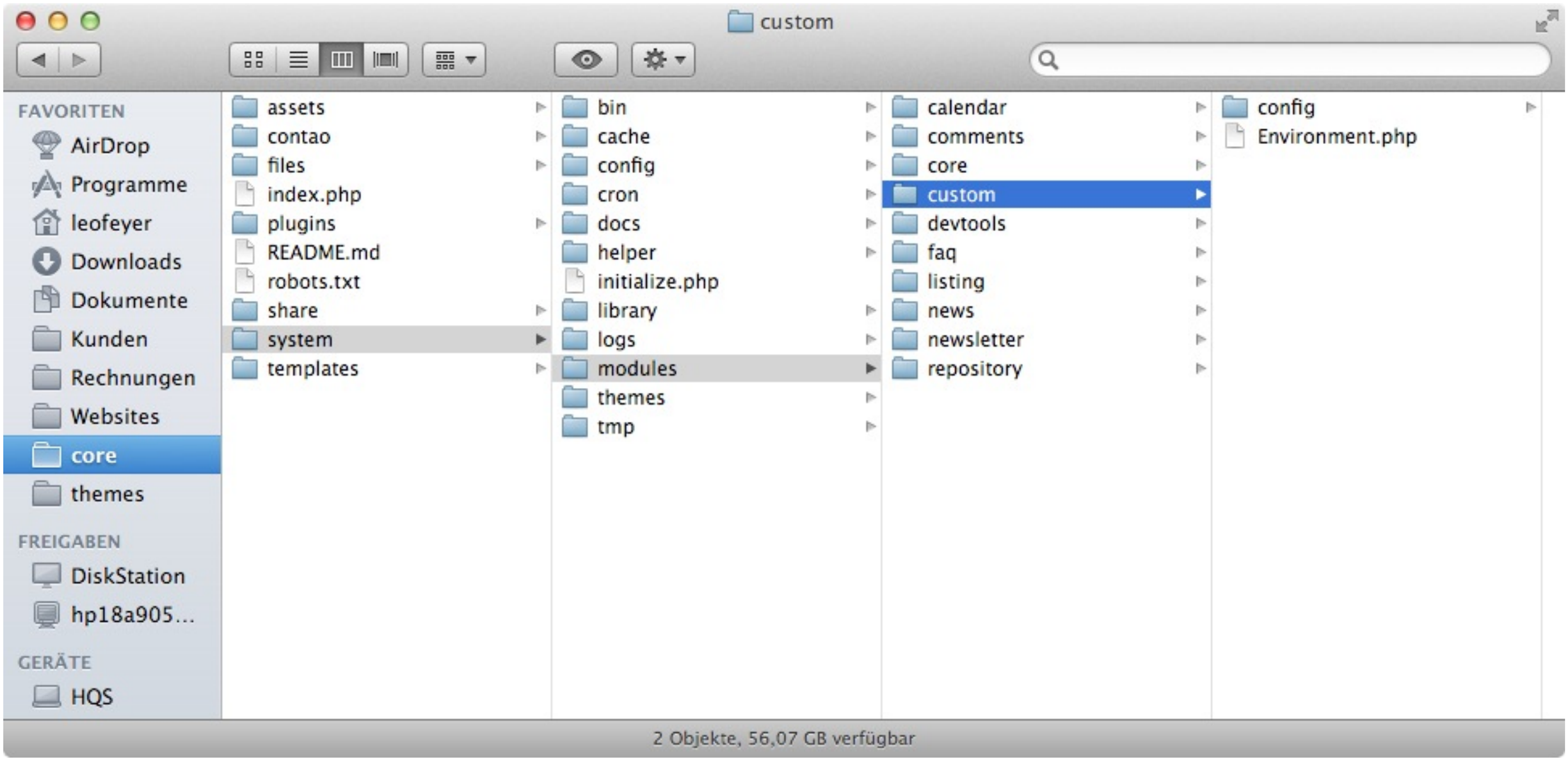
```php
/**
 * Contao Open Source CMS
 *
 * Copyright (C) 2005-2012 Leo Feyer
 *
 * @package Faq
 * @link    http://www.contao.org
 * @license http://www.gnu.org/licenses/lgpl-3.0.html LGPL
 */


/**
 * Register the classes
 */
ClassLoader::addClasses(array
(
    // Models
    'Contao\\FaqCategoryModel' => 'system/modules/faq/models/FaqCategoryModel.php',
    'Contao\\FaqModel'         => 'system/modules/faq/models/FaqModel.php',

    // Modules
    'Contao\\ModuleFaq'        => 'system/modules/faq/modules/ModuleFaq.php',
    'Contao\\ModuleFaqList'    => 'system/modules/faq/modules/ModuleFaqList.php',
    'Contao\\ModuleFaqPage'    => 'system/modules/faq/modules/ModuleFaqPage.php',
    'Contao\\ModuleFaqReader'  => 'system/modules/faq/modules/ModuleFaqReader.php',
));


/**
 * Register the templates
 */
TemplateLoader::addFiles(array
(
    'mod_faqlist'   => 'system/modules/faq/templates',
    'mod_faqpage'   => 'system/modules/faq/templates',
    'mod_faqreader' => 'system/modules/faq/templates',
));
```

- Namespaces
  - Contao 3 nutzt Namespaces, läuft aber im globalen Namespace
- Was heißt das?
  - Oberstes Ziel: Rückwärtskompatibilität
  - Bestehende Erweiterungen laufen im globalen Namespace
  - Daher keine Umstellung auf Namespaces möglich
- Warum dann trotzdem „namespace Contao“?
  - Der Namespace ermöglicht das Überschreiben aller (!) Core-Klassen
  - Der Autoloader erstellt einen Alias der gefundenen Klassen aus dem Namespace im globalen Namensraum, mit dem die Applikation arbeitet
  - Wird eine gleichlautende Klasse in einem eigenen Namespace gefunden, arbeitet Contao automatisch mit dieser anstatt mit der Core-Klasse

- Ein Beispiel
  - Die index.php greift auf die Klasse „Environment“ zu
  - Die Klasse „Environment“ existiert nicht im globalen Namespace
  - Der Autoloader findet die Klasse im Namenspace „Contao“
  - Er erstellt einen Alias der Klasse „Contao\Environment“ im globalen Namespace
  - Die Applikation arbeitet mit „Environment“ und nicht mit „Contao\Environment“

- Eigene Environment-Klasse
  - Erstellen einer eigenen Environment-Klasse
  - Diese liegt im Namespace „Custom“ (Custom\Environment)
  - Der Autoloader findet die Klasse nun im Namespace „Custom“
  - Der Alias im globalen Namespace zeigt jetzt auf „Custom\Environment“
  - Die index.php verwendet weiterhin einfach nur die Klasse „Environment“, ohne zu wissen, dass diese jetzt „Custom\Environment“ ist

- Eigene Klasse im Modul „custom“



Environment.php

Last Saved: 14.05.12 11:04:44
File Path ▾ : ~/Sites/contao/core/system/modules/custom/Environment.php

Environment.php | (no symbol selected)

```php
<?php

/**
 * Namespace
 */
namespace Custom;


/**
 * Class Environment
 */
class Environment extends \Contao\Environment
{

}
```

16 | 1 | HTML | Unicode (UTF-8) | Unix (LF) | 128 / 11 / 16

Autoload-Creator – Contao | Welcome to Music Academy

localhost/contao/core/contao/main.php?do=autoload

Contao Open Source CMS 3.0.beta1

Benutzer k.jones :: Frontend-Vorschau :: Startseite :: Abmelden

Backend-Module

- **Inhalte**
  - Artikel
  - Nachrichten
  - Events
  - FAQ
  - Newsletter
  - Formulargenerator
  - Kommentare
- **Layout**
  - Themes
  - Seitenstruktur
  - Templates
- **Benutzerverwaltung**
  - Mitglieder
  - Mitgliedergruppen
  - Benutzer
  - Benutzergruppen
- **System**
  - Dateiverwaltung
  - System-Log
  - Einstellungen
  - Systemwartung
  - Wiederherstellen
  - Erweiterungskatalog
  - Erweiterungsverwaltung
- **Entwickler-Tools**
  - Extension-Creator

Autoload-Creator

Zurück

Die autoload.php-Dateien automatisch erstellen

Verfügbare Module

- [ ] *Alle auswählen*
- [ ] calendar
- [ ] comments
- [ ] core
- [x] custom
- [ ] devtools
- [ ] faq
- [ ] listing
- [ ] news
- [ ] newsletter
- [ ] repository

Optionen

- [ ] Bestehende Autoload-Dateien überschreiben
- [ ] Die IDE-Kompatibilitätsdatei aktualisieren

Hier können Sie die *config/autoload.php*-Dateien, die für Contao 3 gebraucht werden und die Klassen und Templates zum Autoloader hinzufügen, automatisch erstellen. Falls in einer PHP-Klasse der *namespace*-Befehl gefunden wird, wird der Namensraum ebenfalls hinzugefügt.

Weiter

autoload.php

Last Saved: 14.05.12 11:04:32
File Path ▾ : ~/Sites/contao/core/system/modules/custom/config/autoload.php

autoload.php | (no symbol selected)

```php
<?php

/**
 * Contao Open Source CMS
 *
 * Copyright (C) 2005-2012 Leo Feyer
 *
 * @package Custom
 * @link    http://www.contao.org
 * @license http://www.gnu.org/licenses/lgpl-3.0.html LGPL
 */


/**
 * Register the namespaces
 */
ClassLoader::addNamespaces(array
(
    'Custom',
));


/**
 * Register the classes
 */
ClassLoader::addClasses(array
(
    // Custom
    'Custom\\Environment' => 'system/modules/custom/Environment.php',
));

```

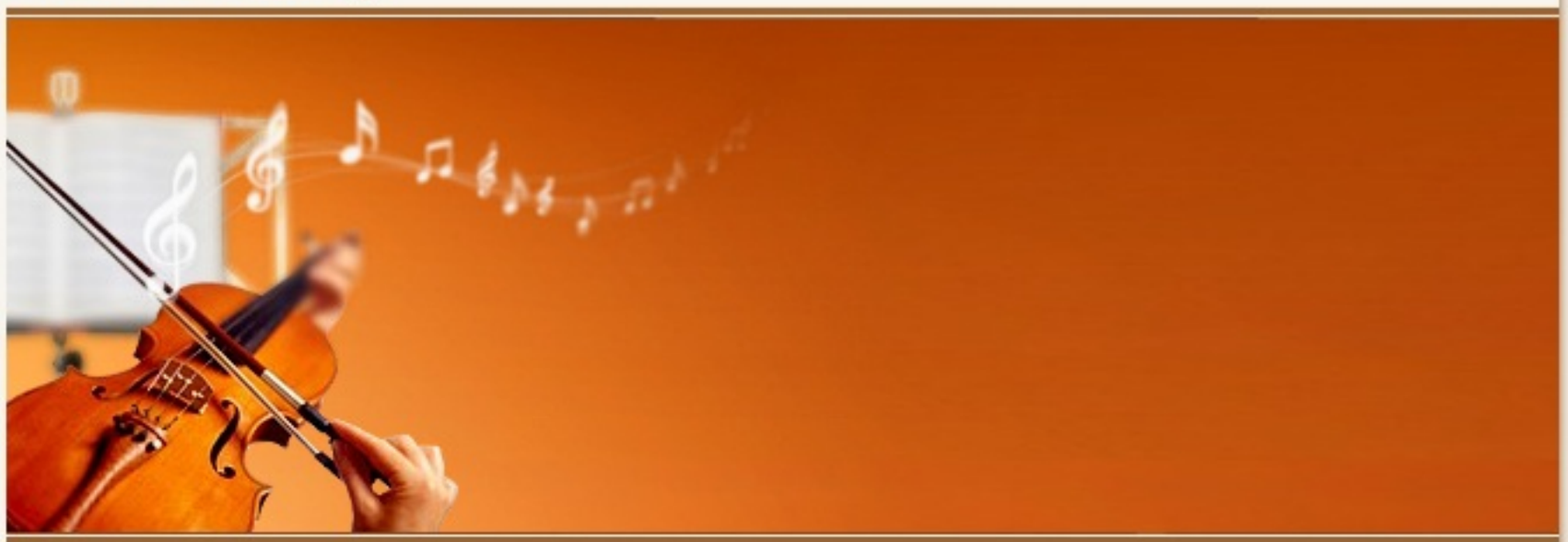
1 | 1 | HTML | Unicode (UTF-8) | Unix (LF) | 433 / 50 / 31

Einstellungen – Contao Open | Welcome to Music Academy

localhost/contao/core/en/

Music Academy

- **Home**
- **The academy**
  - News
  - Calendar
  - Impressions
  - Teachers
- **Courses**
- **Services**
  - Search
  - Navigation
  - Login



Music Academy > Welcome to Music Academy

Front End Login

Username

k.jones

Contao demo website

A content management system is usually divided into two areas: the back end (administration area) and the front end (website). If you are reading this, you are currently viewing the front end. To log in to the back end, simply add */contao*

Contao debug information | Execution time: 0.6689 seconds | Memory usage: 3.3 MB | Database queries: 24

```
Array
(
    [classes_aliased] => Array
        (
            [0] => Input (Contao\Input)
            [1] => RequestToken (Contao\RequestToken)
            [2] => System (Contao\System)
            [3] => Frontend (Contao\Frontend)
            [4] => Controller (Contao\Controller)
            [5] => FrontendUser (Contao\FrontendUser)
            [6] => User (Contao\User)
            [7] => Session (Contao\Session)
            [8] => Database (Contao\Database)
            [9] => Database_Mysqli (Contao\Database_Mysqli)
            [10] => Environment (Custom\Environment)
            [11] => SessionModel (Contao\SessionModel)
            [12] => Model (Contao\Model)
            [13] => Model_QueryBuilder (Contao\Model_QueryBuilder)
            [14] => DcaExtractor (Contao\DcaExtractor)
            [15] => Database_Installer (Contao\Database_Installer)
            [16] => File (Contao\File)
```

Teil 3:

Objektorientierter Zugriff auf Datensätze

- Mit Models lassen sich
  - Datensätze finden
  - Datensätze durchlaufen und lesen
  - Datensätze verändern
  - Datensätze löschen
  - Datensätze neu erstellen

- Objektorientierte Verwendung
  - Ein Datensatz entspricht einem Objekt (Model)
  - Mehrere Datensätze werden als mehrere einzelne Objekte (Models) in einer Objekt-Kollektion (Model_Collection) verwaltet

- Neuer Datensatz
  - ```
    $user = new UserModel();
    $user->name = 'Leo Feyer';
    $user->location = 'Bad Soden';
    $user->save();
    ```
- Äquivalent zu
  - ```
    $db->prepare('INSERT INTO tl_user SET name=?, location=?')
    ->execute('Leo Feyer', 'Bad Soden');
    ```

- Datensatz ändern
  - ```
    $user = UserModel::findById(4);
    $user->location = 'Bad Soden';
    $user->save();
    ```
- Äquivalent zu
  - ```
    $db->prepare('UPDATE tl_user SET location=? WHERE id=4')
       ->execute('Bad Soden');
    ```

- Datensatz löschen
  - `UserModel::findById(4)->delete();`
- Äquivalent zu
  - `$db->execute('DELETE FROM tl_user WHERE id=4');`

- Mehrere Datensätze finden
  -
```
$users = UserModel::findBy('location', 'Bad Soden');

while ($users->next())
{
   echo $users->name;
}
```
- Äquivalent zu
  -
```
$users = db->prepare('SELECT * FROM tl_user WHERE location=?')
            ->execute('Bad Soden');

while ($users->next())
{
   echo $users->name;
}
```

- Interface
  - Um rückwärtskompatibel zu sein, müssen sich Models genauso verhalten wie die Database-Klassen und implementieren daher auch dieselbe Schnittstelle
- Keine Validierung
  - Keine Prüfung der Daten des Objekts vor dem Speichern
  - Die Validierung erfolgt in Contao auf Controller-Ebene (Widget)

- PHP-Magic
  - `UserModel::findBy('location', 'Bad Soden');`
  - `UserModel::findByLocation('Bad Soden');`
- Alles kann, nichts muss
  - Die Verwendung von Models ist komplett optional
  - Es kann auch weiterhin direkt mit der Database-Klasse gearbeitet werden

- Unterstütze Methoden
  - Finde einen Datensatz
    - `findByPk()` Findet einen Datensatz anhand des Primärschlüssels
    - `findOneBy()` Findet einen Datensatz anhand der übergebenen Parameter
    - `findOneBy*()` Findet einen Datensatz anhand eines bestimmten Feldes
    - `findByIdOrAlias()` Findet einen Datensatz anhand seiner ID oder seines Alias
  - Finde mehrere Datensätze
    - `findBy()` Findet Datensätze anhand der übergebenen Parameter
    - `findBy*()` Findet Datensätze anhand eines bestimmten Feldes
    - `findAll()` Findet alle Datensätze
  - Zähle die Ergebnisse
    - `countBy()` Zählt die Ergebnisse einer `findBy()`-Abfrage
    - `countAll()` Zählt die Ergebnisse einer `findAll()`-Abfrage

- Komplexeres Beispiel
  - WHERE mit mehreren Spalten (String oder Array)

```
➔ public static function findPublishedById($intId)
  {
      $t = static::$strTable;
      $arrColumns = array("$t.id=?");

      // Zeige nur veröffentlichte Artikel
      if (!BE_USER_LOGGED_IN)
      {
          $time = time();
          $arrColumns[] = "($t.start='' OR $t.start<$time) AND
                ($t.stop='' OR $t.stop>$time) AND $t.published=1";
      }

      return static::findOneBy($arrColumns, $intId);
  }
```

- Komplexeres Beispiel
  - FIND_IN_SET()

```
➔ public static function findMultipleByIds($arrIds)
  {
     if (!is_array($arrIds) || empty($arrIds))
     {
        return null;
     }

     $arrIds = implode(',', array_map('intval', $arrIds));

     $t = static::$strTable;
     $db = \Database::getInstance();

     return static::findBy
     (
        array("$t.id IN(" . $arrIds . ")"),
        null,
        array('order'=>$db->findInSet("$t.id", $arrIds))
     );
  }
```

- Relationen
  - Die Seite zu einem Artikel finden

```
➔ $article = ArticleModel::findById(12);
  $page = $article->getRelated('pid');
```

  - $page ist jetzt ein Objekt vom Typ PageModel
- Eager oder lazy loading
  - Wird ein in Beziehung stehendes Objekt „eagerly“ geladen, erstellt der QueryBuilder automatisch ein JOIN-Query und lädt beide Objekte in einer Datenbank-Abfrage
  - Wird ein in Beziehung stehendes Objekt „lazy“ geladen, wird es erst auf Anfrage in einer separaten Datenbank-Abfrage nachgeladen
- Einheitliches Interface
  - Die Methode getRelated() lädt das in Beziehung stehende Objekt, egal ob die Daten bereits „eagerly“ geladen wurden, oder noch „lazy“ nachgeladen werden müssen

```
            'label'                   => &$GLOBALS['TL_LANG']['tl_article']['alias'],
            'exclude'                 => true,
            'inputType'               => 'text',
            'search'                  => true,
            'eval'                    => array('rgxp'=>'alias', 'doNotCopy'=>true, 'spaceToUnderscore'=>true, 'maxlength'=>128, 't
            'save_callback' => array
            (
                array('tl_article', 'generateAlias')
            ),
            'sql'                     => "varbinary(128) NOT NULL default ''"

        ),
        'author' => array
        (
            'label'                   => &$GLOBALS['TL_LANG']['tl_article']['author'],
            'default'                 => $this->User->id,
            'exclude'                 => true,
            'inputType'               => 'select',
            'foreignKey'              => 'tl_user.name',
            'eval'                    => array('doNotCopy'=>true, 'mandatory'=>true, 'chosen'=>true, 'includeBlankOption'=>true, '
            'sql'                     => "int(10) unsigned NOT NULL default '0'",
            'relation'                => array('type'=>'hasOne', 'load'=>'eager')
        ),
        'inColumn' => array
        (
            'label'                   => &$GLOBALS['TL_LANG']['tl_article']['inColumn'],
            'exclude'                 => true,
            'default'                 => 'main',
            'inputType'               => 'select',
            'options_callback'        => array('tl_article', 'getActivePageSections'),
            'reference'               => &$GLOBALS['TL_LANG']['tl_article'],
            'sql'                     => "varchar(32) NOT NULL default ''"
        ),
        'keywords' => array
        (
            'label'                   => &$GLOBALS['TL_LANG']['tl_article']['keywords'],
            'exclude'                 => true,
            'inputType'               => 'textarea',
```

- Was passiert im Hintergrund?
  - Zu jedem Artikel wird automatisch der Autor geladen
  - Der Model_QueryBuilder erstellt automatisch eine JOIN-Abfrage
    - → SELECT ... FROM tl_article LEFT JOIN tl_user ON ...
  - `$article->getRelated('author')` gibt das Benutzer-Model zurück
  - Beim Aufruf von `getRelated()` muss keine DB-Abfrage ausgeführt werden

- Das dazugehörige Seiten-Model
  - Die übergeordnete Seite ist „lazy related“
  - Sie wird nicht automatisch per JOIN zum Result-Set hinzugefügt
  - `$article->getRelated('pid')` gibt das Seiten-Model zurück
  - Beim Aufruf von `getRelated()` wird im Hintergrund eine DB-Abfrage ausgeführt
    - → SELECT * FROM tl_page WHERE id=?

Teil 4

API-Änderungen in Contao 3

- Statische Methoden
  - Deklaration nicht-objektbezogener Methoden als statisch
- Contao 2
  - ```php
    $this->import('String');
    echo $this->String->substr($string, 24);
    ```
- Contao 3
  - ```php
    echo String::substr($string, 24);
    ```
- Rückwärtskompatibilität
  - Auch die Variante aus Contao 2 funktioniert in Contao 3
  - Alle Änderungen sind optional und müssen nicht übernommen werden

- Betroffene Klassen
  - Cache — Nicht-persistente Speicherung von Daten
  - Encryption — Verschlüsselung
  - Environment — Umgebungsvariablen auslesen ($_SERVER)
  - Input — Benutzereingaben auslesen ($_GET, $_POST, $_COOKIE)
  - RequestToken — Request-Token erstellen und prüfen
  - Search — Suchindex erstellen und Suchabfragen ausführen
  - String — String-Manipulation
- Neue hinzugekommen
  - Idna — Internationale Domainnamen
  - Image — Bilder verkleinern
  - Message — Meldungen im Backend ausgeben
  - Validator — Prüfung auf bestimmte Formate anhand regulärer Ausdrücke

Contao Open Source CMS

api3.contao.org

Contao Open Source CMS | API Documentation | Reports

Back to top

Contao

API Documentation

NAMESPACES

- Contao

PACKAGES

- Library
- Models

REPORTS

- Errors 0
- Markers
  - todo 4
  - fixme 2
- Deprecated elements 46

NAVIGATION
- Cache.php
- Controller.php
- Encryption.php
- Environment.php
- Folder.php
- Input.php
- RequestToken.php
- Search.php
- String.php
- System.php
- Template.php

Back to top

■ \ Deprecated elements

Cache.php 7

Type	Line	Description
deprecated	103	Cache is now a static class
deprecated	111	Cache is now a static class
deprecated	119	Use Cache::has() instead
deprecated	134	Use Cache::get() instead
deprecated	154	Use Cache::set() instead
deprecated	168	Use Cache::remove() instead
deprecated	181	Cache is now a static class

Controller.php 4

Type	Line	Description
deprecated	3223	Use Image::resize() instead
deprecated	3241	Use Image::get() instead
deprecated	3260	Specify 'datepicker'=>true in your DCA file instead
deprecated	3273	Use String::parseSimpleTokens() instead

Encryption.php 3

INHERITED | PROTECTED | PRIVATE | DEPRECATED

METHODS

Check whether a key is set

has()

Return a cache entry

get()

Add a cache entry

set()

Remove a cache entry

remove()

PROPERTIES

DEPRECATED

C \ Contao \ Cache

A static class to store non-persistent data

This class functions as a global cache container where you can store data that is reused by the application. The cache content is not persisted, so once the process is completed, the data is gone.

Usage:

```
public function getResult()
{
    if (!Cache::has('result'))
    {
        Cache::set('result') = $this->complexMethod();
    }
    return Cache::get('result');
}
```

Package	Library
Author	Leo Feyer <https://github.com/leofeyer>
Copyright	Leo Feyer 2011-2012

Methods

Check whether a key is set

```
has(string $strKey) : boolean
```

- Wegfall der database.sql
  - Die database.sql-Dateien werden nicht mehr benötigt
  - Datenbank-Definitionen können direkt im DCA eingegeben werden
- Rückwärtskompatibilität
  - database.sql-Dateien funktionieren weiterhin auch in Contao 3
  - Die Umstellung ist wie bei allen anderen Änderungen optional

```php

/**
 * Table tl_lock
 */
$GLOBALS['TL_DCA']['tl_lock'] = array
(

    // Config
    'config' => array
    (
        'sql' => array
        (
            'keys' => array
            (
                'id' => 'primary',
                'name' => 'index'
            )
        )
    ),

    // Fields
    'fields' => array
    (
        'id' => array
        (
            'sql'                     => "int(10) unsigned NOT NULL auto_increment"
        ),
        'tstamp' => array
        (
            'sql'                     => "int(10) unsigned NOT NULL default '0'"
        ),
        'name' => array
        (
            'sql'                     => "varchar(64) NOT NULL default ''"
        )
    )
);

```

- Eigene Edit-Buttons
  - Neue Sektion im DCA zum Hinzufügen eigener Buttons
  - `$GLOBALS['TL_DCA'][…]['edit']['buttons_callback']`
- Beispiel: Aliase generieren
  - Datei system/modules/core/dca/tl_page.php
  - Der Callback `addAliasButton()` fügt das Markup und die Logik hinzu

Seitenstruktur – Contao Ope | Welcome to Music Academy

localhost/contao/core/contao/main.php?do=page&act=select&rt=1f7f68cda78432ab365a582ec9edd04f

Contao Open Source CMS 3.0.beta1

Benutzer k.jones :: Frontend-Vorschau :: Startseite :: Abmelden

Backend-Module

- **Inhalte**
  - Artikel
  - Nachrichten
  - Events
  - FAQ
  - Newsletter
  - Formulargenerator
  - Kommentare
- **Layout**
  - Themes
  - Seitenstruktur
  - Templates
- **Benutzerverwaltung**
  - Mitglieder
  - Mitgliedergruppen
  - Benutzer
  - Benutzergruppen
- **System**
  - Dateiverwaltung
  - System-Log
  - Einstellungen
  - Systemwartung
  - Wiederherstellen
  - Erweiterungskatalog
  - Erweiterungsverwaltung
- **Entwickler-Tools**
  - Extension-Creator

Seitenstruktur

Suchen: Seitenalias =

Zurück

Alle auswählen

- Contao Open Source CMS
  - **Music Academy**
    - Page not found
    - Access denied
    - Home
    - The academy
      - News
      - Calendar
      - Events
      - Impressions
      - Teachers
    - Courses
    - Services

Löschen | Verschieben | Kopieren | Überschreiben | Bearbeiten | Aliase generieren

Teil 5

Sonstige Änderungen im Kurzüberblick

- Datenbank-Adapter
  - Zukünftig stehen nur noch MySQL und MySQLi zur Verfügung
  - Die anderen Adapter wurden entfernt, da sich in den letzten 6 Jahren niemand gefunden hat, der diese vervollständigt hätte
- Minütliche automatisierte Aufgaben
  - Der Command Scheduler wurde um das Intervall „minütlich“ erweitert
  - Achtung: Erfolgt der Trigger nicht über einen Cronjob, kann die minütliche Ausführung nicht garantiert werden!
- Inaktive Erweiterungen
  - Deaktivierung durch Hinzufügen einer Datei namens .skip
  - Kein Speichern der inaktiven Erweiterungen in der Konfigurationsdatei mehr, damit diese nicht doppelt geladen werden muss

Fragen

Alle Unklarheiten beseitigt?

The End

Vielen Dank für die Aufmerksamkeit