

# Leicht zu erweitern

Ein großer Vorteil von Typolight ist, dass die Erweiterbarkeit des Systems von Anfang an bedacht und entsprechend umgesetzt wurde. *Von Leo Feyer*

## Info

### Auf einen Blick

- »Der Workshop zeigt an einigen Beispielen, wie Sie das Content-Management-System Typolight updatesicher erweitern können.
- »Sie erfahren, wie das System aufgebaut ist und welche Punkte bei individuellen Erweiterungen zu beachten sind.

### Das brauchen Sie

- »Eine Typolight-Installation. Das im Workshop bearbeitete Beispiel-Modul finden Sie unter [www.phpjournal.de](http://www.phpjournal.de) in der Rubrik *Aktuelles Heft*.

»Fast alles lässt sich aus einem Modulordner heraus ändern, ohne dass dafür irgendwelche Core-Dateien angepasst werden müssten. Darüber hinaus bieten so genannte Hooks Schnittstellen zu allen wichtigen Core-Funktionen. Wenn Ihre Erweiterung also der Anpassung einer Core-Klasse bedarf, und es noch keinen entsprechenden Hook gibt, fragen Sie einfach im Typolight-Forum einen neuen nach.

Neben dem Einsatz von Layoutvorlagen bietet Typolight noch eine ganze Reihe weiterer Anpassungsmöglichkeiten. Umfangreiche Änderungen kapseln Sie am besten in einem eigenen Modulordner, damit Sie den Überblick behalten und Ihre Arbeit beim nächsten Update nicht versehentlich überschrieben wird. Kleinere Anpassungen können Sie dagegen in der Datei *system/config/dcaconfig.php* updatesicher speichern.

### »Die Konfigurationsdatei *dcaconfig.php*«

Typolight speichert alle Informationen über Tabellen und Eingabefelder in so genannten Data Container Arrays (DCA). Für jede Tabelle in der Datenbank muss ein solches DCA vorliegen, damit Typolight weiß, wie die Daten dieser Tabellen zu behandeln sind (zum Beispiel welche Eingaben in einem Feld erlaubt sind).

Die einzelnen Data Container Arrays sind in separate Dateien gekapselt, die sich im *dca*-Verzeichnis jedes Modulordners befinden (zum Beispiel *system/modules/backend/dca*). Zur Laufzeit werden die verschiedenen DCAs der einzelnen Modulordner zu einem großen Array kombiniert, anhand dessen Typolight Auflistungen, Strukturen und Eingabemasken im Backend generiert.

Wie schon erwähnt, erstellt Typolight zur Laufzeit aus den DCA-Dateien der einzelnen Erweiterungen ein großes Array. Die Konfigurationsdatei *dcaconfig.php* wird dabei ganz am Schluss eingelesen, so dass Sie darin jede Ein-

stellung dauerhaft überschreiben können, ohne die Dateien der Erweiterung selbst bearbeiten zu müssen.

### »Pflichtfelder erstellen«

Beim Anlegen eines neuen Mitglieds in der Benutzerverwaltung müssen Sie immer die Felder Vorname, Nachname und E-Mail-Adresse ausfüllen. Andernfalls bricht Typolight die Verarbeitung der Eingaben mit einer Fehlermeldung ab, da diese drei Felder im Data Container Array als Pflichtfelder definiert wurden:

```
$GLOBALS['TL_DCA']['tl_member']['fields']['firstname']['mandatory'] = true;
```

Nehmen wir an, Sie betreiben eine kommerzielle Webseite mit Typolight und wollen Ihre Kunden als Mitglieder anlegen. Da Ihre Kunden ausschließlich Firmen sind, benötigen Sie die Pflichtfelder *Vor- und Nachname* nicht und möchten stattdessen das Eingabefeld *Firma* zum Pflichtfeld machen. Um das zu erreichen, muss das Data Container Array der Tabelle *tl\_member* entsprechend angepasst werden. Öffnen Sie also die Konfigurationsdatei *dcaconfig.php* und fügen Sie folgende Anweisungen ein:

```
$GLOBALS['TL_DCA']['tl_member']['fields']['firstname']['mandatory'] = false;
```

```
$GLOBALS['TL_DCA']['tl_member']['fields']['lastname']['mandatory'] = false;
```

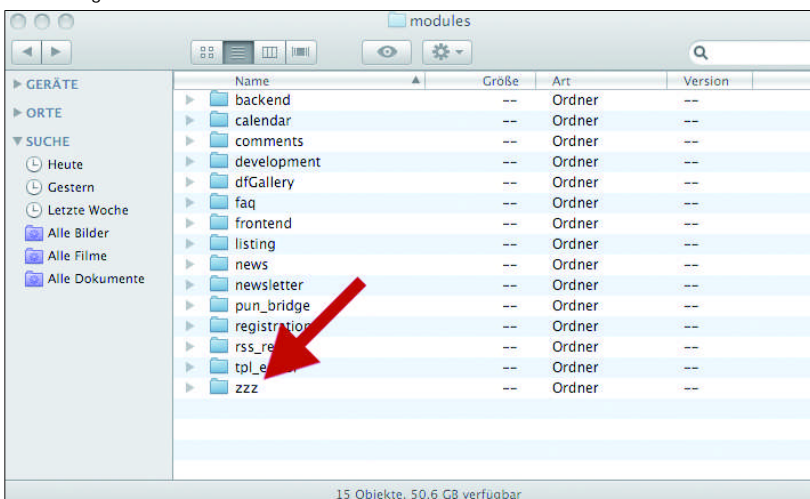
```
$GLOBALS['TL_DCA']['tl_member']['fields']['company']['mandatory'] = true;
```

### »Eingabeprüfung aktivieren«

Bestimmt ist Ihnen schon aufgefallen, dass Typolight bei bestimmten Feldern genau weiß, welche Eingaben erlaubt sind und Sie zum Beispiel mit der Fehlermeldung *Geben Sie nur Zahlen ein* oder *Das Feld darf keine Leerzeichen enthalten* auf falsche Eingaben hinweist. Diese Eingabeprüfung wird ebenfalls im Data Container Array einer Tabelle erfasst und kann von Ihnen beliebig an Ihre eigenen Bedürfnisse angepasst werden. Erweitern wir also das Eingabefeld *Firma* nicht nur zum Pflichtfeld machen, sondern zusätzlich dafür sorgen, dass dort nur Zahlen und Buchstaben, aber keine Sonderzeichen eingegeben werden können. Auch diese Änderung erfassen Sie in der Datei *dcaconfig.php*:

```
$GLOBALS['TL_DCA']['tl_member']['fields']['company']['rgxp'] = 'alnum';
```

Einen eigenen Modulordner am Ende einfügen.





»Inhaltselemente abschalten«

In der *dcaconfig.php* können Sie nicht nur das Data Container Array überschreiben, sondern auch alle anderen Parameter der globalen Konfiguration. In der Konfiguration wird zum Beispiel festgelegt, welche Inhaltselemente zur Verfügung stehen:

```
/**
 * Content elements
 */
$GLOBALS['TL_CTE'] = array
(
    'texts' => array
    (
        'headline' => 'ContentHeadline',
        'text' => 'ContentText',
        'html' => 'ContentHtml',
        'list' => 'ContentList',
        'table' => 'ContentTable',
        'accordion' => 'ContentAccordion',
        'code' => 'ContentCode'
    ),
    'links' => array
    (
        'hyperlink' => 'ContentHyperlink',
        'toplink' => 'ContentToplink'
    ),
    'images' => array
    (
        'image' => 'ContentImage',
        'gallery' => 'ContentGallery'
    ),
    'files' => array
    (
        'download' => 'ContentDownload',
        'downloads' => 'ContentDownloads'
    ),
    'includes' => array
    (
        'alias' => 'ContentAlias',
        'teaser' => 'ContentTeaser',
        'form' => 'Form',
        'module' => 'ContentModule'
    )
);
```

Buch-Tipp

# Das offizielle Typolight-Handbuch

Ambitionierte Anwendern und Entwicklern gibt das Buch einen Einblick in das Innenleben des CMS Typolight.

Kein anderer könnte Ihnen Typolight besser erklären als der Entwickler des preisgekrönten Web-CMS selbst, Leo Feyer. In diesem offiziellen Handbuch führt er Sie von der Installation über die Administration bis zur Erweiterung von Typolight. Unterwegs lernen Sie alles, was Sie für Aufbau und Pflege einer Website mit dem erfolgreichen Web-CMS wissen müssen.

Sie beginnen mit der Installation auf einer lokalen Testumgebung oder einem Live-Server und machen sich anschließend mit Administration und Funktionsweise von Typolight vertraut. Sie lernen, mit Inhaltselementen zu arbeiten, den Typolight-internen Dateimanager und den Formulargenerator zu nutzen sowie Ihre Website mit Frontend-Modulen und Core-Erweiterungen auszubauen. Ein gesondertes Administrationskapitel behandelt die Systemwartung und Benutzerverwaltung, ein Kapitel für Entwickler erläutert die wichtigsten Klassen des Typolight-Frameworks und beschreibt, wie Sie eigene Module erstellen.

[mb]



Autor Leo Feyer  
 Verlag Addison-Wesley  
 ISBN 978-3-8273-2686-7  
 Preis 29,95 Euro  
 288 Seiten

```
),
    'includes' => array
    (
        'alias' => 'ContentAlias',
        'teaser' => 'ContentTeaser',
        'form' => 'Form',
        'module' => 'ContentModule'
    )
);
```

Um bestimmte Inhaltselemente zu deaktivieren,

müssen Sie lediglich die entsprechenden Einträge aus dem Konfigurationsarray entfernen. Nehmen wir zum Beispiel an, Sie möchten die Verwendung von Include-Elementen generell verbieten, dann können Sie diese mit Hilfe folgender Anweisung abschalten:

```
unset($GLOBALS['TL_CTE']['includes']);
```

Unter Umständen möchten Sie die Includes-Elemente auch nur für Ihre Editoren abschalten, selbst aber schon damit arbeiten. In diesem Fall soll nur Administratoren der Zugriff auf die Elemente erlaubt sein:

```
if (!$this->User->isAdmin())
{
    unset($GLOBALS['TL_CTE']['includes']);
}
```

Wie Sie sehen, sind die Möglichkeiten der *dcaconfig.php* nahezu unbegrenzt. Und da alle Änderungen auch bei einem Update erhalten bleiben, ist sie das ideale Instrument für kleinere Anpassungen der Typolight-Konfiguration.

»Anpassungen in einem Modulordner kapseln«

Natürlich können Sie nicht nur die Konfiguration vorhandener Felder anpassen, sondern auch eigene Eingabefelder hinzufügen. In so einem Fall ist es sinnvoll, die Änderungen in einem eigenen Modulordner zu kapseln, da Sie dort noch mehr Anpassungsmöglichkeiten haben als in der *dcaconfig.php*.

```
30 /**
31  * Table tl_article
32  */
33 $GLOBALS['TL_DCA']['tl_article'] = array
34 (
35     // Config
36     'config' => array
37     (
38         'dataContainer' => 'Table',
39         'table' => 'tl_page',
40         'ctable' => array('tl_content'),
41         'switchToEdit' => true,
42         'enableVersioning' => true,
43         'onload_callback' => array
44         (
45             array('tl_article', 'checkPermission')
46         )
47     ),
48     // List
49     'list' => array
50     (
51         'sorting' => array
52         (
53             'mode' => 6,
54             'fields' => array('published DESC', 'title', 'author'),
55             'paste_button_callback' => array('tl_article', 'pasteArticle')
56         ),
57         'label' => array
58         (
59             'fields' => array('title', 'inColumn'),
60             'format' => '%s <span style="color:#333333; padding-left:3px;">[&#x]</span>',
61             'label_callback' => array('tl_article', 'addImage')
62         ),
63         'global_operations' => array
64         (
65             'all' => array
66             (
67                 'label' => &GLOBALS['TL_LANG']['MSC']['all'],
68                 'href' => 'act-select',
69                 'class' => 'header_edit_all',
70                 'attributes' => 'onclick="Backend.getScrollOffset();"
71             )
72         ),
73         'toggleNodes' => array
74         (
75             'label' => &GLOBALS['TL_LANG']['MSC']['toggleNodes'],
76             'href' => '&amp;ptg=all',
77             'class' => 'header_toggle'
78         )
79     )
80 );
```

Auszug des Data Container Arrays der Tabelle tl\_article.

Im vorhergehenden Abschnitt haben Sie gelernt, wie Sie das Eingabefeld *Firma* zu einem Pflichtfeld machen, so dass Sie Ihre Kunden als Mitglieder erfassen können. Nehmen wir nun an, dass jeder Kunde in der Buchhaltung bereits eine Kundennummer hat, die ebenfalls gespeichert werden soll. Das Feld *Kundennummer* gibt es standardmäßig nicht in der Tabelle *tl\_member*, daher müssen Sie es neu erstellen.

Legen Sie als Erstes einen neuen Unterordner im Verzeichnis *system/modules* an und beachten Sie, dass Typolight die Modulordner nacheinander in alphabetischer Reihenfolge einliest. Wenn Sie also beispielsweise die Nachrichten-Erweiterung anpassen möchten, darf Ihr Modulordner nicht *anpassungen* heißen, da er sonst vor *news* verarbeitet würde und die Nachrichten-Konfiguration Ihre Änderungen überschriebe anstatt umgekehrt. Am besten nennen Sie Ihren Modulordner daher zzz.

Bitte füllen Sie das Feld "Kundennummer" aus!

Bitte geben Sie die 8-stellige Kundennummer ein.

Firma

Hier können Sie einen Firmennamen eingeben.

Straße

Bitte geben Sie den Namen der Straße und die Hausnummer ein.

Postleitzahl

Bitte geben Sie die Postleitzahl ein.

Stadt

Bitte geben Sie den Namen der Stadt ein.

Staat

Bitte geben Sie den Namen des Staates ein.

Land

Bitte wählen Sie ein Land.

Das neue Eingabefeld im Backend.

Als Nächstes muss das neue Eingabefeld in die Tabelle *tl\_member* eingefügt werden, so dass es das Typolight-Install-Tool erkennt und bei der Aktualisierung berücksichtigt. Dazu müssen Sie nichts weiter tun, als den Unterordner *config* anzulegen und dort eine Datei namens *database.sql* mit folgendem Inhalt zu erstellen:

```
CREATE TABLE 'tl_member' (
```

**Update database tables**

⊘ The database is not up to date!

Please note that this update assistant has only been tested with MySQL and MySQLi databases. If you are using a different database (e.g. Oracle), you might have to install/update your database manually. In this case, please go to folder **system/modules** and search all its subfolders for files called **dca/database.sql**.

**Add new columns**

ALTER TABLE `tl\_member` ADD `kundennummer` varchar(8) NOT NULL default '';

Update database

Aktualisierung der Datenbank über das Install-Tool.

```
' kundennummer' varchar(8) NOT NULL
default ''
) ENGINE=MYISAM DEFAULT
CHARSET=utf8;
```

Genau wie aus den einzelnen DCA-Dateien zur Laufzeit ein großes Konfigurationsarray erstellt wird, werden auch die einzelnen SQL-Dateien vom Install-Tool zu einem Array zusammengefasst und dann mit der Datenbank verglichen. Das Install-Tool wird Ihnen daher jetzt anbieten, das Feld *kundennummer* anzulegen.

» Data Container Array erweitern«  
Das neue Eingabefeld existiert zwar nun in der Datenbank, allerdings hat Typolight von seiner Existenz noch nichts mitbekommen. Sie müssen zuerst das Data Container Array der Tabelle *tl\_member* erweitern, damit die Core-Engine weiß, wie das Feld im Backend dargestellt werden soll. Legen Sie dazu einen weiteren Unterordner *namens dca* an und erstellen Sie darin die PHP-Datei *tl\_member.php* mit folgendem Inhalt:

```
// Modify palette
$GLOBALS['TL_DCA']['tl_member']
['palettes']['default'] = str_replace('company',
'kundennummer, company',
$GLOBALS['TL_DCA']['tl_member']
['palettes']['default']);
// Add field
$GLOBALS['TL_DCA']['tl_member']
['fields']['kundennummer'] = array(
'label' =>
```

```
&$GLOBALS['TL_LANG']['tl_member']
['kundennummer'],
'exclude' => true,
'inputType' => 'text',
'eval' => array(
'mandatory' => true,
'rgxp' => 'digit',
'maxlength' => 8
);
```

Die erste Anweisung sorgt dafür, dass das Eingabefeld für die Kundennummer im Backend angezeigt wird. Die zweite Anweisung beschreibt das Feld:

- n label: verknüpft das Feld mit einem Label aus dem Spracharray.
- n exclude: sorgt dafür, dass das Feld in einem Benutzerprofil deaktiviert werden kann.
- n inputType: legt fest, dass es sich bei dem Feld um ein Textfeld handelt.
- n mandatory: macht das Feld zu einem Pflichtfeld.
- n rgxp: sorgt dafür, dass nur Zahlen eingegeben werden können.
- n maxLength: sorgt dafür, dass maximal 8 Zeichen eingegeben werden können.

Als Letztes müssen Sie noch die Feldbezeichnung und eine kurze Beschreibung eingeben. Legen Sie dazu einen weiteren Unterordner namens *languages* an und erstellen Sie darin weitere Unterordner für jede gewünschte Sprache. Für unser Beispiel reicht ein Verzeichnis namens *de*, in dem die deutsche Übersetzung gespeichert wird. Legen Sie dort eine PHP-Datei namens *tl\_member.php* mit folgendem Inhalt an:

```
$GLOBALS['TL_LANG']['tl_member']
['kundennummer'] = array(
'Kundennummer', 'Bitte geben Sie die 8-stellige Kundennummer ein.');
```

Damit haben Sie alle Bestandteile des neuen Eingabefeldes zusammen und können das Ergebnis Ihrer Arbeit im Backend-Modul *Mitglieder* bewundern.

» Den Rich Text Editor anpassen«  
In diesem Abschnitt geht es darum, wie Sie die Konfiguration von TinyMCE so anpassen, dass Ihre Änderungen bei einem Update erhalten bleiben. Die Konfigurationsmöglichkeiten des Rich-Text-Editors

## Info Das Typolight-Framework

Typolight liegt ein MVC-Framework zugrunde, das speziell für die Anforderungen des Systems programmiert wurde.

Es weicht an manchen Stellen von der klassischen MVC-Architektur zugunsten einer individuellen Lösung ab (zum Beispiel bei der Erstellung von Frontend-URLs), sollte Ihnen als Entwickler aber keine Probleme bereiten, wenn Sie mit dem Aufbau und der Funktionsweise von MVC-Frameworks vertraut sind. Kern des Typolight-Frameworks sind verschiedene Klassen

(Bibliotheken), die sich im Ordner *system/libraries* befinden und diverse Aufgaben wie zum Beispiel das Erstellen von Dateien, das Verarbeiten von Benutzereingaben oder das Interagieren mit der Datenbank kapseln. Möglichen Sicherheitslücken, die durch Cross-Site-Scripting oder SQLInjection ausgenutzt werden könnten, wird innerhalb des Frameworks wirksam vorgebeugt, daher sollten Sie es auch in Ihren eigenen Erweiterungen stets benutzen. Die vollständige Dokumentation der API finden Sie auf der Projektwebseite.

api.typolight.org



sind auf der TinyMCE-Projektwebseite ([wiki.moxiecode.com/index.php/TinyMCE:Index](http://wiki.moxiecode.com/index.php/TinyMCE:Index)) ausführlich beschrieben. Typolight verwendet eine abgespeckte Version des Rich-Text-Editors, die an das System und die Anforderungen der Barrierefreiheit angepasst wurde. Abgespeckt bedeutet, dass nicht alle vorhandenen TinyMCE-Plugins installiert sind. Bei Bedarf können Sie die fehlenden Plugins nachrüsten, indem Sie sie herunterladen und in den Typolight-Ordner `plugins/tinyMCE/plugins` verschieben.

In Typolight können Sie verschiedene Konfigurationsdateien für TinyMCE anlegen und so den Rich-Text-Editor an die Erfordernisse des jeweiligen Inhaltselements oder Moduls, in dem er verwendet wird, anpassen. So stehen beispielsweise im Editor eines Flash- Inhaltes wesentlich weniger Funktionen zur Verfügung als im Editor eines Inhaltselements, weil Flash nur sehr wenige HTML-Tags unterstützt und es keinen Sinn machen würde, Formatierungen anzubieten, die später nicht verarbeitet werden können. Die TinyMCE-Konfigurationsdateien befinden sich wie alle anderen Konfigurationsdateien im Ordner `system/config`. Die Standardkonfiguration für das Text-Element ist in der Datei `tinyMCE.php` gespeichert.

Um eine eigene Konfigurationsdatei zu erstellen, kopieren Sie am besten eine vorhandene Datei und benennen diese einfach um. Passen Sie den Inhalt an Ihre Bedürfnisse an und übertragen Sie die Datei dann auf den Server.

Danach müssen Sie Typolight nur noch mitteilen, für welche Eingabefelder er Ihre eigene Konfigurationsdatei laden soll. Diese Information wird ebenfalls im Data Container Array gespeichert und kann daher ganz leicht in der `dcaconfig.php` überschrieben werden. Um beispielsweise die Datei `tinyCustom.php` dauerhaft allen Text-Elementen zuzuweisen, müssen Sie lediglich folgende Zeile einfügen:

```
$GLOBALS['TL_DCA']['tl_content']
['fields']['text']['eval']['rte'] =
'tinyCustom';
```

»Der Modul-Creator«

Alle Typolight-Erweiterungen sind nach einem ähnlichen Schema aufgebaut, das auf einer einheitlichen Verzeichnisstruktur und bestimmten Dateien beruht. So gibt es zum Beispiel in jeder Er-

```
35 <script type="text/javascript" src="<?php echo $this->base; ?>plugins/tinyMCE/tiny_mce_gzip.js"></script>
36 <script type="text/javascript">
37 <!--><![CDATA[</-->
38 tinyMCE.GZ.init({
39     plugins : "advimage,autosave,contextmenu,emotions,save,searchreplace,spellchecker,style,table,template,typolinks,xhtmlxtras",
40     themes : "advanced",
41     languages : "<?php echo $this->language; ?>",
42     disk_cache : false,
43     debug : false
44 });
45 </--></script>
46 </script>
47 <script type="text/javascript">
48 <!--><![CDATA[</-->
49 tinyMCE.init({
50     mode : "exact",
51     height : "300",
52     language : "<?php echo $this->language; ?>",
53     elements : "<?php echo $this->rtefields; ?>",
54     dialog_type : "modal",
55     inline_styles : true,
56     force_br_newlines : true,
57     force_p_newlines : false,
58     remove_linebreaks : false,
59     force_hex_style_colors : true,
60     apply_source_formatting : true,
61     convert_fonts_to_spans : true,
62     font_size_style_values : "8pt,10pt,12pt,14pt,18pt,24pt,36pt",
63     doctype : "<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">",
64     document_base_url : "<?php echo $this->base; ?>",
65     entities : "5B,nbsp,60,lt,62,gt",
66     cleanup_on_startup : true,
67     save_enablewhendirty : true,
68     save_on_tinymce_forms : true,
69     save_callback : "TinyCallback.cleanXHTML",
70     advimage_update_dimensions_onchange : false,
71     external_image_list_url : "../plugins/tinyMCE/plugins/typolinks/typoimages.php",
72     template_external_list_url : "../plugins/tinyMCE/plugins/typolinks/typotemplates.php",
73     plugins : "advimage,autosave,contextmenu,emotions,save,searchreplace,spellchecker,style,table,typolinks,template,xhtmlxtras",
74     spellchecker_languages : "English=en,Danish=da,Dutch=nl,Finnish=fi,French=fr,German=de,Italian=it,Polish=pl,Portuguese=pt,Spani",
75     content_css : "../basic.css,.../system/themes/tiny_mce.css",
76     event_elements : "a,div,h1,h2,h3,h4,h5,h6,img,p,span",
77     extended_valid_elements : "q[cite=class|title]",
78     theme : "advanced",
79     theme_advanced_resizing : true,
80     theme_advanced_resize_horizontal : false,
81     theme_advanced_resizing_use_cookie : false,
82     theme_advanced_toolbar_location : "top",
83     theme_advanced_toolbar_align : "left",
84     theme_advanced_statusbar_location : "bottom",
```

So sieht die Konfigurationsdatei des Rich Text Editors von Typolight aus.

weiterung den Unterordner `config` mit der Datei `config.php`, in der die Typolight-Konfiguration angepasst werden kann.

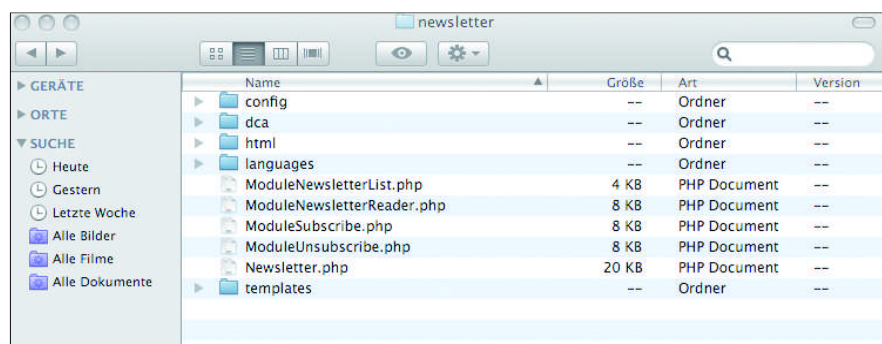
Damit Sie die Verzeichnisse und Dateien einer neuen Erweiterung nicht jedes Mal manuell anlegen müssen, gibt es den Modul-Creator, der das für Sie übernimmt. Der Modul-Creator ist Teil der Developer-Erweiterung, die speziell für Entwickler gedacht ist und neben dem Modul-Creator noch ein Tool zum Auffinden fehlender Bezeichnungen in Typolight-Übersetzungen enthält.

Sie finden den Modul-Creator in der Backend-Navigation in der Gruppe `System`. Klicken Sie auf den Link `Modul-Creator` und anschließend auf den Link `Neues Modul`, um eine neue Erweiterung zu erstellen. Die folgenden Eingaben sind erforderlich:

- n TITEL: Der Titel einer Erweiterung wird nur in der Backend-Übersicht verwendet.
- n ORDNERNAME: Hier legen Sie den Namen des Verzeichnisses fest, in dem die Dateien der Erweiterung gespeichert werden. Die Erweiterung wird als Unterordner im Verzeichnis `system/modules` gespeichert, daher sollten Sie darauf

achten, dass der gewählte Name möglichst eindeutig und noch nicht vergeben ist.

- n AUTOR: Geben Sie hier Ihren Vor- und Nachnamen sowie Ihre E-Mail-Adresse ein.
- n COPYRIGHT: Geben Sie hier einen Copyright-Vermerk ein (zum Beispiel *Ihr Name 2008*).
- n PAKET: Hier können Sie Ihre Erweiterung mit einem Paketnamen versehen, der später von Dokumentationsprogrammen wie dem `phpDocumentor5` ausgewertet werden kann. Der Paketname sollte aussagekräftig sein und keine Leerzeichen enthalten.
- n LIZENZ: Geben Sie hier den Namen der Lizenz ein, unter der Sie Ihr Modul veröffentlichen möchten (zum Beispiel GPL, LGPL oder MIT-Lizenz). Damit haben Sie den allgemeinen Teil Ihrer Erweiterung erfolgreich konfiguriert. Die folgenden Schritte hängen ganz davon ab, was für eine Art von Erweiterung Sie schreiben möchten. Sie können sowohl Backend- als auch Frontend-Module hinzufügen. Dazu sind folgende Eingaben erforderlich:
- n EIN BACKENDMODUL HINZUF?GEN: Wählen Sie diese Option, wenn Sie Dateien für das Backend hinzufügen möchten (zum Beispiel für ein neues Eingabefeld).
- n BACKEND-KLASSEN: Geben Sie eine durch Kommata getrennte Liste der Backend-Klassen ein, die der Modul-Creator für Sie erstellen soll.
- n BACKEND-TABELLEN: Geben Sie eine durch Kommata getrennte Liste von Tabellennamen an, die Sie für Ihre Erweiterung benötigen.
- n BACKEND-TEMPLATES: Geben Sie eine durch Kommata getrennte Liste von Templates an, die Sie in Ihrer Erweiterung verwenden möchten.
- n EIN FRONTENDMODUL HINZUF?GEN: Wählen Sie diese Option, wenn Sie Dateien für das Frontend hinzufügen möchten (zum Beispiel für



Typische Verzeichnisstruktur einer Erweiterung.

ein neues Modul oder Inhaltselement).

- n FRONTEND-KLASSEN: Geben Sie eine durch Kommata getrennte Liste der Frontend-Klassen ein, die der Modul-Creator für Sie erstellen soll.
- n FRONTEND-TABELLEN: Geben Sie eine durch Kommata getrennte Liste von Tabellennamen an, die Sie für Ihre Erweiterung benötigen.
- n FRONTEND-TEMPLATES: Geben Sie eine durch Kommata getrennte Liste von Templates an, die Sie in Ihrer Erweiterung verwenden möchten.
- n EIN SPRACHPAKET ERSTELLEN: Wählen Sie diese Option, wenn Sie zu Ihrer Erweiterung ein oder mehrere Sprachpakete erstellen möchten.
- n SPRACHEN: Geben Sie eine durch Kommata getrennte Liste mit Sprachen an. Sprachen werden über ihr primäres Subtag nach ISO 639-1 erfasst, also zum Beispiel *de* für deutsch.

Die nächsten Schritte sollen anhand eines Beispielmoduls, mit dem Sie eigene Formulare erstellen und deren Eingaben in der Datenbank speichern können, demonstriert werden. Die dazu benötigten Verzeichnisse und Dateien legt der Modul-Creator automatisch für Sie an. Nach dem Speichern der Änderungen gelangen Sie zurück zur Übersicht. Dort ist unsere Erweiterung nun gelistet, jedoch wurden noch keinerlei Dateien oder Ordner angelegt. Damit das geschieht, müssen Sie auf der Übersichtsseite des Modul-Creators auf das entsprechende Navigationsicon klicken. Den nun folgenden Hinweis sollten Sie genau lesen. Er macht

Das neue Modul im Backend konfigurieren.

Sie darauf aufmerksam, dass eventuell früher erstellte Dateien während des Erstellungsprozesses überschrieben werden. Sichern Sie daher die erstellten Dateien auf Ihrem lokalen Rechner. Wechseln Sie danach in das Verzeichnis *system/modules* und sehen Sie sich an, welche Dateien und Ordner der Modul-Creator angelegt hat.

»Das Modul konfigurieren«

Öffnen Sie zunächst die Datei *config/config.php* und registrieren Sie das Modul, damit Typo3 es finden kann. Fügen Sie dazu folgende Zeile ein:

```
$GLOBALS['FE_MOD'] ['application']
['custom_form'] = 'ModuleCustomForm';
```

Anschließend müssen Sie Typo3 mitteilen, welche zusätzlichen Datenbank-Felder Sie benötigen. Diese Information ist in der Datei *config/database.sql* gespeichert. Das Beispielm modul soll ein Formular enthalten, mit dem Mitglieder ihre Teilnahme an einer Veranstaltung bestätigen oder ablehnen können. Also benötigen wir ein Feld für den Namen, ein Feld für die E-Mail-Adresse, ein Feld für die Zu- oder Absage und ein optionales Kommentarfeld. Dementsprechend lautet der SQL-Code:

```
CREATE TABLE
'tl_custom_form' (
'id' int(10) unsigned NOT
NULL auto_increment,
'tstamp' int(10) unsigned
NOT NULL default '0',
'name' varchar(64) NOT NULL
default '',
'email' varchar(128) NOT NULL
default '',
'attending' char(1) NOT NULL default
'',
'comments' text NULL,
PRIMARY KEY ('id')
) ENGINE=MyISAM DEFAULT
CHARSET=utf8;
```

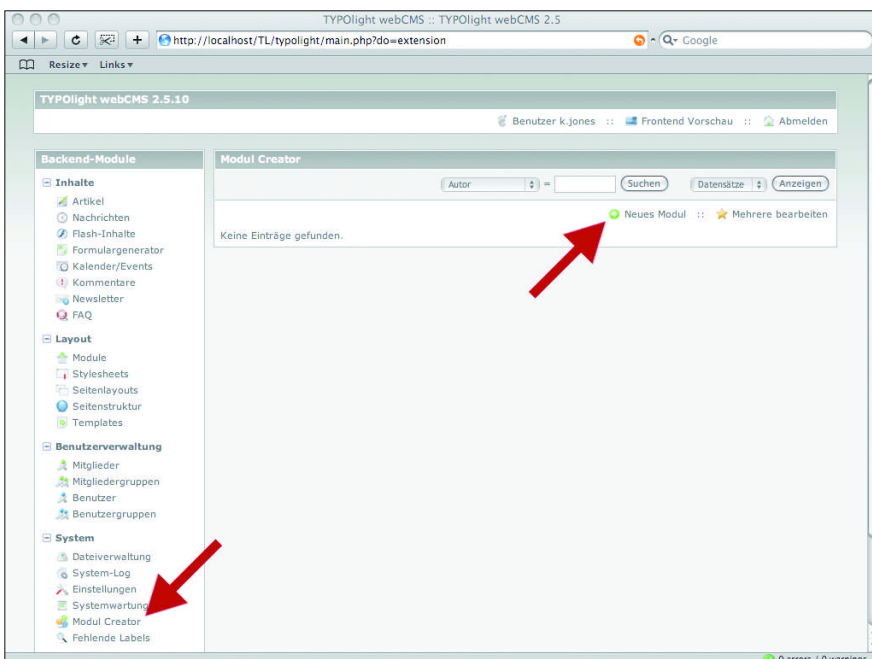
Die Konfiguration des Moduls ist damit abgeschlossen. Sie können nun die Datenbank mit dem Install-Tool aktualisieren und das Modul über die Modulverwaltung aufrufen. Um die Eingabefelder wie gewohnt im Backend bearbeiten zu können, müssen Sie noch festlegen, welche speziell zu dem Modul gehören. Diese Information speichert Typo3 im Data Container Array, den Sie aus den vorherigen Abschnitten kennen. Öffnen Sie die Datei *dca/tl\_module.php* und fügen Sie folgende Zeile ein:

```
$GLOBALS['TL_DCA']['tl_module']
['palettes']['custom_form'] =
'name, type, headline; guests,
protected; align, space, cssID';
```

Als letztes sollten Sie dem Modul noch einen Namen geben, damit im Backend nicht nur *custom\_form* angezeigt wird. Öffnen Sie dazu die Sprachdatei *languages/de/modules.php* und fügen Sie folgende Zeile ein:

```
$GLOBALS['TL_LANG'] ['FMD']
['custom_form'] = array('Eigenes
Formular', 'Ein eigenes Formular
erstellen.');
```

Jetzt können Sie Ihr Modul in der Modulverwaltung konfigurieren. Natürlich enthält das Modul noch



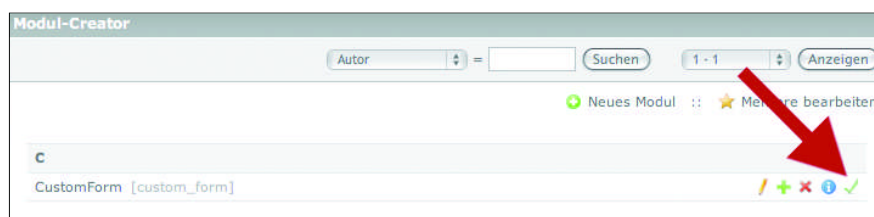
Ein neues Modul mit dem Modul-Creator erstellen.



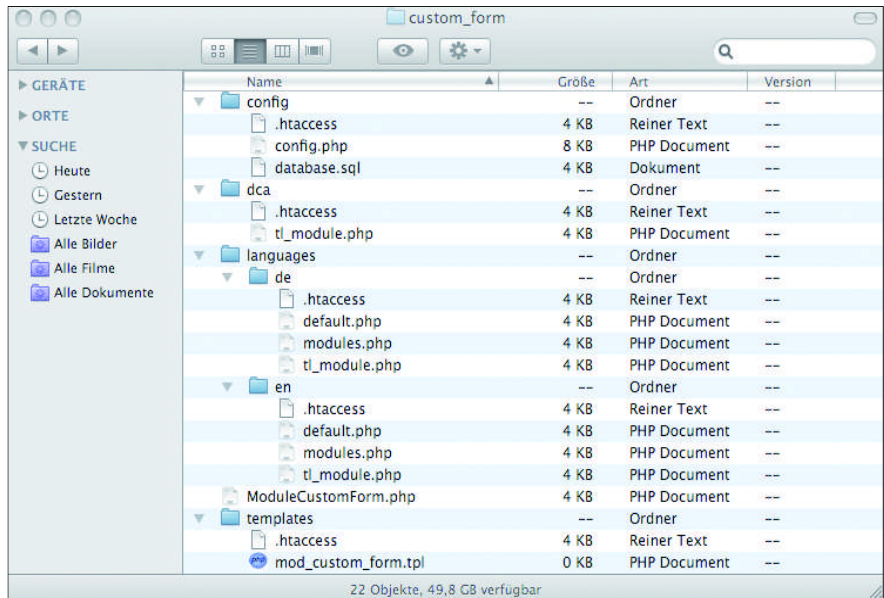
keine Ausgabe, weil Sie ja noch keine Programmlogik hinzugefügt haben. Typolight kapselt den Code in Klassen, die ebenfalls vom Module-Creator angelegt werden. Öffnen Sie die Datei *ModuleCustomForm.php* und fügen Sie Ihren Code in die Methode *compile()* ein:

```
protected function compile()
{
// Eingaben verarbeiten
if ($this->Input-
>post(' FORM_SUBMIT' ) ==
' custom_form' )
{
$this->Database->prepare
(
"INSERT INTO
tl_custom_form
SET
name=?,
email=?,
attending=?,
comments=?,
tstamp=?"
)
->execute
(
$this->Input->post(' name' ),
$this->Input->post(' email' ),
$this->Input->post(' attending' ),
$this->Input->post(' comments' ),
time()
);
$this->reload();
}
// Template-Variablen
$this->Template->action = $this-
>Environment->request;
}
```

Diese relativ einfache Methode nimmt die Benutzereingaben entgegen, speichert sie in der Datenbank und lädt dann die Seite neu, um ein doppeltes Abschicken zu verhindern. Ihre eigene Programmlogik wird sehr wahrscheinlich deutlich komplizierter ausfallen, weil Sie zum Beispiel die E-Mail-Adresse prüfen oder bereits erhaltene Zu- oder Absagen bei einem erneuten Abschicken aktualisieren möchten. Zu guter Letzt fehlt nur noch das Template, das den eigentlichen HTML-Code enthält. Sie finden es unter *templates/mod\_custom\_form.tpl*:



Die Dateien einer Erweiterung erstellen.



Vom Modul-Creator angelegte Strukturen.

```
<!-- indexer::stop -->
<div class="<?php echo $this->class;
?> block">
<h1><?php echo $this->headline;
?></h1>
<form action="<?php echo $this-
>action; ?>" method="post">
<div class="formbody">
<input type="hidden"
name="FORM_SUBMIT"
value="custom_form" />
<p>
<label for="cf_name">Name</label ><br />
<input type="text" name="name"
id="cf_name" class="text" value=""
/>
</p>
<p>
<label for="cf_email">E-Mail -
Adresse</label ><br />
<input type="text" name="email"
id="cf_email" class="text" value="@"
/>
</p>
<p>
<label
for="cf_comments">Kommentar</label >
<br />
<textarea name="comments"
```

```
id="cf_comments" class="textarea"
rows="6" cols="30">
</textarea>
</p>
<p>
<input type="radio" name="attending"
id="cf_attending_1" class="radio"
value="" />
<label for="cf_attending_1">Ich
nehme teil</label ><br />
<input type="radio" name="attending"
id="cf_attending_0" class="radio"
value="1" />
<label for="cf_attending_0">Ich
nehme nicht teil</label ><br />
</p>
<p>
<input type="submit" class="submit"
value="Abschicken" />
</p>
</div>
</form>
</div>
<!-- indexer::continue -->
```

Ihr eigenes Modul ist nun einsatzbereit.

»Typolight-Framework«

Bestimmt sind Ihnen in der Klasse *ModuleCustomForm* die Konstrukte *\$this->Input->post()* oder *\$this->Database->prepare()* aufgefallen. Sie referenzieren bestimmte Funktionen des Typolight-Frameworks. Es ist zu empfehlen, auch Ihre eigenen Erweiterungen darauf aufzubauen, um so eventuelle Konflikte oder Sicherheitslücken von vornherein zu minimieren. [mb]

Wenn Sie mehr über das CMS Typolight erfahren wollen, lesen Sie das Buch »Das offizielle Typolight-Handbuch« vom Autor dieses Artikels, Leo Feyer, erschienen im Verlag Addison-Wesley.