Franz Ripfel

TYPO3 Extensions entwickeln

Teil 2: Aufbau und Inhalt von Extensions analysieren und verstehen

Extensions sind das Lebenselixier von TYPO3. Jeder kann TYPO3 mit Hilfe einer eigenen Extension anpassen oder erweitern – und das ist gar nicht so schwer. Diese dreiteilige Artikelserie bietet einen Einstieg in die Programmierung eigener Extensions. Teil 2 bringt Ihnen den Aufbau und den Inhalt von Extension näher.

Teil 1 (T3N Nr. 10)	Extensions kennen lernen, Kickstarter lieben lernen
Teil 2 (T3N Nr. 11)	Aufbau und Inhalt von Extensions analysieren und verstehen
Teil 3 (T3N Nr. 12)	TYPO3-API für die eigene Programmierung richtig einsetzen

Im ersten Teil haben Sie die sehr unkomplizierte Erstellung von Extensions mit dem Kickstarter kennengelernt. Als Ergebnis sollten Sie Ihre erste eigene Extension abgespeichert haben, auch wenn diese vielleicht noch nicht genau das Gewünschte leistet. Dieser Teil widmet sich dem Aufbau von Extensions, quasi dem Innenleben. Dabei werden Sie auch die Unterschiede zwischen Extensions für verschiedene Einsatzbereiche kennenlernen.

Extension-Daten in ext_emconf.php

Die Datei „ext_emconf.php“ im Hauptordner Ihrer Extension wird vom Extension Manager benötigt und enthält alle Informationen zur Extension, die für eine Installation respektive Deinstallation notwendig sind. Sie wird beim Speichern der Extension angelegt. Die Informationen sind im Array „$EM_CONF[$_EXTKEY]“ hinterlegt. In vielen Fällen müssen Sie an den vom Extension Kickstarter angelegten Einstellungen keine Änderungen mehr vornehmen. Die Zuständigkeit der einzelnen Felder erklärt sich häufig bereits durch den verwendeten Namen. Wenn Sie diese Datei aus der von Ihnen erstellten oder einer von Ihnen eingesetzten Extension öffnen, werden Sie schnell den Zusammenhang zwischen den gesetzten Werten und den darauf basierenden Auswirkungen erkennen. Aus Platzgründen kann dieser Artikel nur eine besonders spannende Auswahl an Konfigurationsmöglichkeiten vorstellen. Wir werden uns auf diejenigen konzentrieren, die nicht bereits vom Kickstarter konfiguriert wurden.

Als „shy“ markierte Extensions ('shy' = '1') werden vom Extension Manager nur angezeigt, falls das entsprechende Häkchen bei „Display shy extensions“ (ganz oben im Extension Manager) gesetzt ist.

Besonders sensible Extensions werden nur bei explizitier Aktivierung angezeigt.

Abhängigkeiten, Konflikte und Einschränkungen

Die schon seit langem vorhandenen Konfigurationseinstellungen für die Festlegung von Abhängigkeiten und Konflikten zu anderen Extensions sind heutzutage mit Hilfe des Felds „constraints“ sehr genau definierbar. Diese flexiblere Art, Einschränkungen für die Installation einer Extension anzugeben, wurde mit dem TER und dem dazugehörigen neuen Extension Manager für die TYPO3-Version 4.0 eingeführt. Sie können dadurch zusätzlich zu den Extensions auch noch bestimmte Versionen ansprechen (siehe Listing 1).

PHP

```
'constraints' => array(
  'depends' => array(
    'typo3' => '3.8.1-',
    'php' => '4.3.2-5.2.2',
    'cms' => '',
  ),
  'conflicts' => array(
    'badext' => '-1.2.4'
  ),
  'suggests' => array(
    'goodext' => ''
  ),
),
```

Listing 1

Im Beispiel definieren Sie für Ihre Extension die Notwendigkeit von TYPO3 in Version 3.8.1 oder höher, eine PHP-Version von 4.3.2-5.2.2 und die Extension „cms“ in beliebiger Version. Die fiktive Extension „“badext“ wird in den Versionen bis zu 1.2.4 einen Konflikt im Extension Manager hervorrufen, die Extension „goodext“ wird zur Installation empfohlen, aber nicht zwingend gefordert.

Das Feld „priority“ ist eine sehr selten benutzte, aber enorm hilfreiche Einstellung, die über die Installationsreihenfolge der Extensions entscheidet. Nutzen Sie dieses Feld, falls Sie sichergehen wollen, dass von Ihnen getroffene Einstellungen nicht von anderen Extensions überschrieben werden. Mögliche Einstellungen sind „top“, „“ und „bottom“. Zuerst werden alle mit „top“ gekennzeichneten Extensions in der Reihenfolge der Installation geladen (was zuerst installiert wurde, wird auch zuerst geladen), dann folgen alle Extensions ohne Eintrag und am Schluss alle mit der Kennzeichnung „bottom“. Der Extension Manager setzt diese Einstellung um, indem er beim Installieren einer Extension die Reihenfolge der Extension Keys in der Variablen „$TYPO3_CONF_VARS['extList']“ entsprechend manipuliert. Wenn Sie diese Einstellung in einer bereits installierten Extension ändern, müssen Sie diese also zunächst deinstallieren und anschließend noch einmal installieren, damit die Einstellung greift. Die Reihenfolge, in der Extensions eingebunden werden ist deshalb besonders wichtig, weil Einstellungen in der zuletzt geladenen Extension alle früher getroffenen Einstellungen vom Core oder anderen Extensions überschreiben. Ein gutes Anwendungsbeispiel dafür bietet die Extension „abz_eff_template“.

Setzen Sie die Option „clearCacheOnLoad“, wenn es für eine korrekte Funktionsweise erforderlich ist, dass bei der Installation der Extension der Frontend-(Webseiten-)Cache gelöscht wird.

Wichtige reservierte Datei- und Ordnernamen

Auch hier gilt: Analysieren Sie den Aufbau von bekannten Extensions, dies ist zum Verständnis enorm hilfreich. Die Datei

„ext_localconf.php“ bildet eine Erweiterung zur Hauptkonfigurationsdatei „localconf.php“ und wird in der Reihenfolge der Extension-Liste direkt nach dieser eingebunden. Hier können „$TYPO3_CONF_VARS“ gesetzt bzw. überschrieben und benötigte Klassen inkludiert werden. Außerdem werden normale Plugins hier meist vom Extension Kickstarter dem statischen Default-Template hinzugefügt (Listing 2).

PHP

```
t3lib_extMgm::addPItoST43($_EXTKEY,'pi1/
class.tx_abzreferences_pi1.php','_pi1','list_type',1);
```

Listing 2

Solange $TYPO3_CONF_VARS['EXT']['extCache'] = '1' gesetzt ist (Default), werden alle ext_localconf.php-Dateien in der Datei „temp_CACHED_*_ext_localconf.php“ aus Performance-Gründen zusammengefasst und ihre Konfigurationen darüber aufgerufen. Wenn Sie die Datei „ext_localconf.php“ Ihrer Extension manuell geändert haben, ohne die Extension neu zu installieren, klicken Sie den Link „clear cache in typo3conf/“ über dem Logout-Button, um damit die temp_CACHED_*-Dateien zu löschen und die geänderte Konfiguration Ihrer Extension zu laden. Die Datei „ext_localconf.php“ wird bereits vom Extension Kickstarter angelegt, sofern sie gebraucht wird.

Die Datei „ext_tables.php“ bildet analog zur „ext_localconf.php“ die Erweiterung zur Tabellenkonfigurationsdatei „tables.php“ und enthält Anweisungen, welche Plugins und Module von der Extension eingebunden werden sollen. Darüber hinaus befinden sich hier die Angaben zum $TCA (Table Configuration Array). Dieses Array enthält alle nötigen Angaben zu neuen Datenbanktabellen und Feldern und bestimmt damit das Aussehen der Formulare im Backend. Solange $TYPO3_CONF_VARS ['EXT']['extCache'] = '1' gesetzt ist (Default), werden alle ext_tables.php-Dateien in der temp_CACHED_*_ext_tables.php aus Geschwindigkeitsgründen zusammengefasst und darüber aufgerufen. Auch ext_tables.php wird bereits vom Extension Kickstarter angelegt, sofern diese Datei benötigt wird.

Konfigurationen für die Darstellung der Felder in Backend-Formularen können teilweise in die Datei „tca.php“ ausgelagert werden. Wenn Sie bestehende Tabellen erweitern, werden die $TCA-Einstellungen für die neuen Felder in der Datei „ext_tables.php“ hinterlegt. Legen Sie eigene Tabellen neu an, wird die $TCA-Konfiguration für die Tabelle in der Datei „ext_tables.php“ gespeichert, die $TCA-Konfiguration für die Felder aber in der Datei „tca.php“. Die Auslagerung (Listing 3) dient dem Zweck, „ext_tables.php“ bei vielen $TCA-Konfigurationen übersichtlich zu halten.

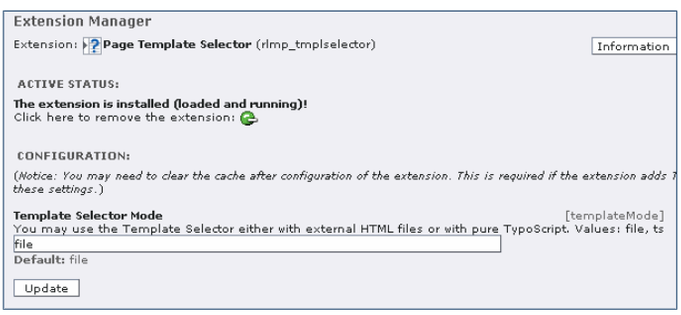
PHP

```
$TCA['tx_yourTable'] = Array (
 'ctrl' => Array (
 [...weitere Definitionen...]
 'dynamicConfigFile' => t3lib_extMgm::extPath($_EXTKEY).'tca.php'
 )
);
```

Listing 3

Falls die Extension neue Datenbanktabellen oder -felder benötigt, werden in der Datei „ext_tables.sql“ die entsprechenden SQL-Befehle hinterlegt. Der Extension Manager greift beim Installieren der Extension darauf zurück, um die Tabellen und Felder anzulegen. Auch bei der Überprüfung der Datenbank auf Konsistenz mit den $TCA-Konfigurationen durch den Aufruf von „compare“ im Install-Tool wird auf diese Informationen zurückgegriffen. Lassen Sie sich nicht durch den Befehl „CREATE TABLE“ irritieren, der auch für neue Felder einer bestehenden Datenbank verwendet wird. Der Extension Manager interpretiert die SQL-Befehle entsprechend. Die Datei „ext_tables.sql“ wird bei Bedarf vom Extension Kickstarter angelegt.

Die Datei „ext_conf_template.txt“ enthält ein Template, das die Konfigurationsmöglichkeiten des Benutzers der Extension bei der Installation definiert. Die Syntax entspricht der des Constant Editors für TypoScript. Aber Achtung: Anders als im Constant Editor kann hier als Kategorie nur „basic“ ausgewählt werden. Die Unterkategorien entsprechen denen des Constant Editors. Eine Angabe der Sortierung wird ignoriert, kann also weggelassen werden. Der Extension Manager erzeugt aus diesen Angaben eine Maske mit den Konfigurationsoptionen, die beim Installieren der Extension ausgewählt werden können und in die Datei „localconf.php“ als serialisiertes Array geschrieben werden. Betrachten wir als Beispiel die Extension „Page Template Selector“ von Robert Lemke (rlmp_tmplselector).



Konfigurationsmöglichkeit im Page Template Selector basierend auf Einträgen in ext_conf_template.txt

Der Benutzer kann bei der Installation (und bei Bedarf auch noch später) auswählen, ob die verschiedenen Templates auf Dateien oder TypoScript basieren sollen. Sie können solcherlei Konfigurationsmöglichkeiten recht einfach zur Verfügung stellen (Listing 4).

TypoScript-Constants zur Exension-Konfiguration

```
# cat=basic//; type=string; label=Template Selector Mode: You may use the Template
Selector either with external HTML files or with pure TypoScript. Values: file, ts
templateMode = file
```

Listing 4

Neben der Zuordnung zu einer Kategorie können Sie den Typ des Felds (type=string) und beschreibenden Text (label=...) definieren. Die hervorgehobene Überschrift wird durch den Doppelpunkt von der tatsächlichen Beschreibung getrennt. Den Namen für die Option können Sie frei vergeben – sinnvoll ist ein Name in englischer Sprache. Als Standardwert ist im Beispiel „file“ vorgegeben, wodurch für die Auswahl der Templates Dateien aus einem in TypoScript zu definierenden Verzeichnis verwendet werden. TYPO3 speichert die Daten serialisiert in „$TYPO3_CONF_VARS“ (Listing 5).

PHP

```
$TYPO3_CONF_VARS['EXT']['extConf']['rlmp_tmplselector'] = 'a:1:{s:
12:"templateMode";s:4:"file";}';
```

Listing 5

Ein sehr umfangreiches Einsatzbeispiel finden Sie in der Extension „“tt_news“ von Rupert Germann. Aufgrund der vielen Konfigurationsmöglichkeiten sind dabei die einzelnen Werte gruppiert dargestellt. Dies können Sie durch eine entsprechende Auswahl vorgefertigter Unterkategorien für Ihre Parameter erreichen – bei-

spielsweise „cat=basic/enable/120“. Diese Unterkategorien für die Strukturierung der Datei „ext_conf_template.txt“ sind hart gecoded in der „t3lib/class.t3lib_tsparser_ext.php“ hinterlegt (Listing 6).

```php
var $subCategories = array(
// Standard categories:
 "enable" => Array("Enable features", "a"),
 "dims" => Array("Dimensions, widths, heights, pixels", "b"),
 "file" => Array("Files", "c"),
 "typo" => Array("Typography", "d"),
 "color" => Array("Colors", "e"),
 "links" => Array("Links and targets", "f"),
 "language" => Array("Language specific constants", "g"),
);
```

Listing 6

Innerhalb der Extension können Sie auf dieses serialisierte Array zugreifen und die enthaltenen Werte entsprechend berücksichtigen. Falls Sie an den Eingaben vor dem Speichern in das serialisierte Array Änderungen vornehmen wollen, können Sie einen dafür vorgesehenen Hook in der Datei „typo3/mod/tools/em/class.em_index.php“ verwenden (Listing 7).

```php
$TYPO3_CONF_VARS['SC_OPTIONS']['typo3/mod/tools/em/index.php']
['tsStyleConfigForm'][] = Ihre Funktion
```

Listing 7

Bereich für (herkömmliche) Frontend-Plugins (pi*)

Frontend-Plugins und die zugehörigen Dateien liegen innerhalb des Extension-Ordners in den Unterordnern „pi*“. In der Regel werden Sie nur den Ordner „pi1“ haben, da zu den meisten Extensions nur ein Frontend-Plugin gehört. Mehrere sind jedoch kein Problem und können einfach durch die Ziffer nach der Zeichenfolge pi unterschieden werden.

Die wichtigste Datei heißt „class.tx_[extName ohne Unterstrich]_pi1.php“ und enthält den logischen Code für die Erzeugung der Darstellung im Frontend. Standardmäßig erbt sie von der Klasse „tslib/class.tslib_pibase.php“ und wird nur um spezifische Funktionalitäten erweitert. Hier findet die meiste Arbeit für den Entwickler des Frontend-Plugins statt. Der Extension-Kickstarter legt bereits eine funktionierende Grundlage. Mit Hilfe eines unscheinbaren Auswahlfeldes (siehe Teil 1 in T3N Nr. 10) haben Sie sogar einen funktionierenden Code-Vorschlag für die Listen- und Einzelansicht der Datensätze einer Ihrer Tabellen erzeugen lassen.

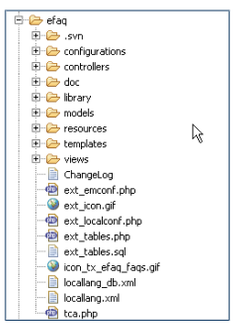
TypoScript-Konfigurationen für die Einbindung von statischen TypoScript-Templates liegen in „static/constants.txt“ und „static/setup.txt“, Sprachlabel in der Datei „locallang.xml“.

Basisklasse tslib/class.tslib_pibase.php ausnutzen

Hier lohnt es sich ganz besonders, vor dem Erstellen des nächsten Plugins die zur Verfügung gestellten Möglichkeiten zu studieren. Es stehen Ihnen viele Optionen zur Erzeugung von Listen, Blätterfunktionen, Suchfunktionen, Lokalisierung und Verlinkung zur Verfügung. Bei Bedarf können diese Methoden auch in Ihrer erbenden pi-Klasse überschrieben werden. Zu empfehlen sind beispielsweise „pi_list_browseresults“, „pi_list_searchBox“ und „pi_wrapInBaseClass“.

Ganz besonders möchte ich Ihnen die Linkfunktionen wie „pi_getPageLink“, „pi_linkToPage“, „pi_linkTP“, „pi_linkTP_keepPIvars“, „pi_linkTP_keepPIvars_url“, „pi_list_linkSingle“ und „pi_openAtagHrefInJSwindow“ ans Herz legen. Durch sie werden Sie in die Lage versetzt, allgemein einsetzbare Extensions zu schreiben. Ihre Links werden auch dann noch funktionieren, wenn die TYPO3-Installation mehrsprachig wird oder durch andere Extensions Übergabeparameter hinzugefügt werden. Ein manuelles Zusammenbauen von Links kann sich im Gegensatz dazu schnell rächen, weil die Links plötzlich nicht mehr zuverlässig greifen. Mit Hilfe der API-Beschreibung und einigen Eigenversuchen werden Sie schnell den jeweils optimalen Einsatz herausfinden.

Bereiche von MVC-Plugins

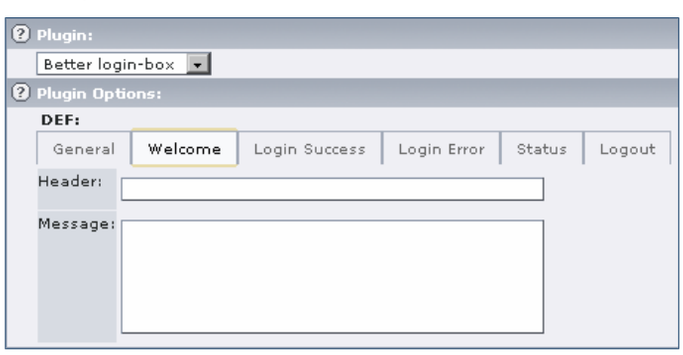


MVC-Plugins sind auf den ersten Blick deutlich komplizierter als herkömmliche pi1-Plugins, helfen aber bei einer sauberen Strukturierung umfangreicher Extensions.

Wie im ersten Teil der Artikelserie bereits angesprochen, lohnt sich für die Programmierprofis unter Ihnen bereits jetzt ein Blick auf die Struktur des zukunftsträchtigen lib/div-Ansatzes [1] des Extension Coordination Teams. Vor allem umfangreichere Extensions sind durch die straffe Aufteilung der Dateien einfacher zu verwalten. Die im TER verfügbare Extension „efaq“ von Elmar Hinz bietet ein umfangreiches Einsatzbeispiel für diesen Ansatz und wartet dabei auch mit einer umfangreichen Dokumentation auf.

Flexforms für Frontend-Plugins

TYPO3 stellt Ihnen ein spezielles XML-Format namens „T3DataStructure“ zur Verfügung. Dieses bietet Ihnen die Möglichkeit, Daten für Ihre Extension strukturiert und hierarchisch abzulegen und dafür fertige TYPO3-Funktionen zu nutzen. Haupteinsatzgebiete für diese Strukturen sind derzeit TemplaVoila und eben Flexforms. Ein anschauliches Beispiel finden Sie in der Extension „newloginbox“.



Durch den Einsatz der Flexforms können Konfigurationsmöglichkeiten sauber und einfach strukturiert werden.

Eine sehr ausführliche Erläuterung von Flexforms bietet die umfangreiche TYPO3-Wiki-Seite zum Thema [2].

Bereich für Backend-Module (mod*)

Backend-Module und zugehörige Dateien liegen innerhalb des Extension-Ordners in den Unterordnern „mod*“. In der Regel werden Sie nur den Ordner „mod1“ haben, da zu den meisten Extensions nur ein Backend-Modul gehört. Mehrere sind jedoch kein Problem und können einfach durch die Ziffer nach der Zeichenfolge „mod“ unterschieden werden.

Die Datei „conf.php“ wird vom Kickstarter fertig angelegt und muss in der Regel nicht mehr angepasst werden. Hier sind Konfigurationsangaben (z. B. für Zugriffsrechte) enthalten, die vom Backend benötigt werden. Falls Sie beim Aufruf eines frisch vom Kickstarter erzeugten Moduls einen Fehler im Backend bekom-

men, weil TYPO3 die Datei „init.php“ nicht findet, sollten Sie versuchen, die Pfade manuell zu ändern (Listing 8).

PHP

```
#define('TYPO3_MOD_PATH', 'ext/abz_references/mod1/');
#$BACK_PATH='../../../';
define('TYPO3_MOD_PATH', '../typo3conf/ext/abz_references/mod1/');
$BACK_PATH='../../../../typo3/';
```

Listing 8

Die Information, wo das Backend-Modul dann angezeigt wird, ist jedoch nicht hier, sondern bereits fertig vom Kickstarter in der Datei „ext_tables.php“ hinterlegt. In der Datei „index.php“ nehmen Sie analog zur pi-Klasse im Frontend die von Ihnen benötigten Änderungen nach der Erzeugung durch den Kickstarter vor.

Wichtig für den Anfang sind dabei die Funktionen „menuConfig“ und „moduleContent“. In „menuConfig“ werden die Elemente für das Auswahlfeld der Funktionalität konfiguriert, in „moduleContent“ wird der entsprechende Code dafür implementiert. Schließlich hinterlegen Sie wiederum in der Datei „locallang.xml“ alle Label, die innerhalb des Backend-Moduls benötigt werden.

Bereich für Services (sv*)

In Ordnern, die mit „sv*“ beginnen, werden Dateien abgelegt, die für Services benötigt werden. So wird vom Kickstarter die Datei „class.tx_[extName ohne Unterstrich]_sv1.php“ erzeugt, die Sie entsprechend Ihren Wünschen anpassen können. Weiterführende Informationen zu TYPO3-Services finden Sie im Buch des Autors oder direkt in der Core-Dokumentation [3].

Der nächste Teil dieser Artikelserie wir Ihnen die umfangreiche TYPO3-API näher bringen. Diese ermöglicht Ihnen, für verschiedenste Aufgaben bereits vorhandene Funktionalitäten zu nutzen und möglichst TYPO3-kompatiblen Code zu schreiben.

Dieser Artikel ist ein Auszug aus:	
Titel:	Das TYPO3-Profihandbuch. Der Leitfaden zu Front- und Backendprogrammierung
Autoren:	Franz Ripfel, Melanie Meyer, Irene Höppner
Verlag/ISBN:	Addison-Wesley/978-3827323224
URL:	http://www.t3buch.de
Preis/Umfang:	49,90 EUR/550 Seiten, gebunden

Links und Literatur

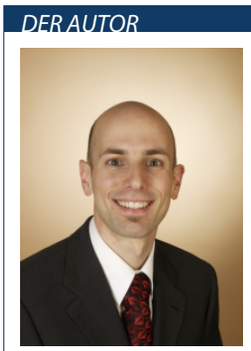
Softlink 1935

[1] MVC-Framework: http://wiki.typo3.org/index.php/MVC_Framework

[2] Extension-Entwicklung mit Flexforms: http://wiki.typo3.org/index.php/Extension_Development,_using_Flexforms

[3] Dokumentation zu TYPO3-Services: http://typo3.org/documentation/document-library/core-documentation/doc_core_services/current/

DER AUTOR



Franz Ripfel befasst sich seit 2003 mit TYPO3 und ist Schritt für Schritt immer tiefer eingestiegen. Er ist Mitbegründer der A.BE.ZET GmbH in München (www.abezet.de), die auf Lösungen für Inter- und Intranet basierend auf TYPO3 spezialisiert ist. Dort leitet er derzeit die Entwicklungsabteilung und ist oft in Sachen Schulungen und Coaching unterwegs.