Franz Ripfel

TYPO3-Extensions entwickeln

Teil 3: TYPO3-API für die eigene Programmierung richtig einsetzen

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 3 zeigt Ihnen, wie Sie die TYPO3-API für die eigene Programmierung richtig nutzen.

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 kennen gelernt. Als Ergebnis sollten Sie dadurch mindestens Ihre erste eigene Extension abgespeichert haben, auch wenn diese vielleicht noch nicht genau das Gewünschte leistet. Im zweiten Teil haben Sie den Aufbau, also quasi das Innenleben, von Extensions sowie die Unterschiede zwischen Extensions für verschiedene Einsatzbereiche kennen gelernt.

Im dritten und letzten Teil stehen mit den TYPO3-APIs die Programmierschnittstellen des freien CMS im Mittelpunkt. Diese bieten Ihnen häufig die Möglichkeit, schon bestehende Funktionalität in Form von Klassen und Methoden für Ihre eigenen Aufgabenstellungen zu nutzen. Nebenbei lernen Sie so besser lesbaren und TYPO3-konformen Code zu schreiben. Der Artikel stellt die wichtigsten Methoden und Eigenschaften sowie Variablen und Konstanten vor, die TYPO3 zur Verfügung stellt.

Verfügbare Konstanten

In einer normalen TYPO3-Umgebung stehen nach dem Einbinden der Datei „init.php" eine ganze Reihe von Informationen bezüglich der Skriptpfade und Datenbankzugriffsdaten zur Verfügung. Da sich diese während der Laufzeit nicht verändern, werden sie durch Konstanten dargestellt. Die Extension „cc_beinfo" [1] von René Fritz bietet eine einfache und schnelle Möglichkeit, die wichtigsten Konstanten im Backend einzusehen. Als Beispiel seien hier die häufig einsetzbaren Konstanten PATH_site, PATH_thisScript und PATH_typo3 genannt.

Ein Großteil der Konstanten ist sowohl im Backend als auch im Frontend verfügbar. Aus Platzgründen werden hier die einzelnen Konstanten nicht alle aufgezählt. Einen guten Überblick liefern hier die Extension „cc_beinfo" und Abschnitt 3.5 (Variables and Constants) der Dokumentation zu TYPO-Core-API [2].

Versuchen Sie immer, diese Konstanten anstatt eigener Pfaddefinitionen zu nutzen, da dann sichergestellt ist, dass die Pfade auf möglichst vielen Systemen und Systemkonfigurationen funktionieren.

Globale Variablen

Dem TYPO3-Entwickler stehen eine ganze Reihe globaler Variablen und Objekte zur Verfügung. Ein Tipp dazu: Greifen Sie auf globale Objekte mit Hilfe von $GLOBALS zu. Darin stehen Ihnen alle globalen Variablen von Haus aus zur Verfügung.

PHP

```
$myExt = $GLOBALS['TYPO3_LOADED_EXT'];
$myConf = $GLOBALS['TYPO3_CONF_VARS'];
```

Listing 1

Die wichtigsten stellen wir hier vor, weitere finden Sie in der Doku zur Core-API [2].

TYPO3_CONF_VARS

Das TYPO3-Konfigurations-Array schlechthin: Die Werte können über das Install Tool gesetzt werden. Dort sehen Sie auch eine Beschreibung der einzelnen Werte. In den TYPO3_CONF_VARS werden auch Konfigurationen für Plugins abgelegt.

TYPO3_DB

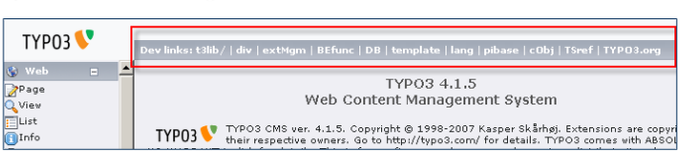
Instanz der DB-Wrapper-Klasse t3lib_db: Dieses Objekt muss für alle Datenbankverbindungen benutzt werden, um Kompatibilität zu anderen Datenbanken wie PostgreSQL, Oracle und MS SQL gewährleisten zu können.

TSFE

Das Objekt TSFE enthält die eigentliche Frontend-Klasse, die für viele Aufgabenstellungen genutzt werden kann. Eine tiefergehende Beschreibung von TYPO3_DB und TSFE finden Sie weiter unten bei der Klassenbeschreibung tslib/tslib_fe.php, class tslib_fe.

Die wichtigsten Klassen für den Extension-Entwickler

Es gibt einige Klassen in TYPO3, um deren Nutzung man als Entwickler von sauber eingebundenen Extensions nicht herumkommt. Diese Klassen stellen Ihnen einen Grundumfang an Funktionalitäten zur Verfügung, auf die Sie in vielen Situationen zurückgreifen können oder müssen. Falls Sie es nicht schon längst getan haben, ist jetzt der richtige Zeitpunkt gekommen, um den Extension Development Evaluator (Extension Key: extdeveval) [3] zu installieren. Dadurch erweitern Sie das Backend um eine Link-Leiste, in der einige dieser Klassen mit Verweisen auf die entsprechende API eingebunden sind.



Die Extension extdeveval erweitert das Backend von TYPO3-Versionen niedriger 4.2 um eine Linkleiste, die auf wichtige Informationen zu ausgewählten Klassen verweist.

Achtung: Zum Zeitpunkt der Artikelerstellung war die aus dem TER erhältliche Extension extdeveval (3.0.0) nicht mit der neuen TYPO3-Version 4.2 kompatibel. Die API-Anzeige scheint in Kombination mit TYPO3 4.2 nicht möglich zu sein, da es den Bereich dafür im Backend schlichtweg nicht mehr gibt. Um die beschriebene Funktionaliät auszuprobieren, greifen Sie also zumindest im Moment am besten auf eine TYPO3-Version 4.1.x zurück.

Damit Sie die Klassen im Verzeichnissystem leichter finden, zeigen die folgenden Überschrift zusätzlichen zum Namen der Klasse auch den Namen der Datei, die die Klassen enthält.

class t3lib_div (class.t3lib_div.php)

Eine der ältesten Klassen in TYPO3, in der viele Funktionalitäten zusammengefasst wurden, die nicht unbedingt TYPO3-spezifisch sind. Deshalb wird die Klasse auch nicht instanziiert, die Funktionen werden statt dessen statisch aufgerufen.

Von der Klasse „t3lib_div" wird überall im TYPO3-Source-Paket fleißig Gebrauch gemacht. Eine kleine Auswahl an oft eingesetzten Funktionen:

PHP

```
// zum Auslesen von Daten aus POST oder GET.
t3lib_div::_GP($var)
// zum Einfügen von Umbrüchen in textbasierten E-Mails.
t3lib_div::breakTextForEmail($str,$implChar="\n",$charWidth=76)
// zum Überprüfen, ob ein Element in einem Array vorkommt.
t3lib_div::inArray($in_array,$item)
// schreibt Inhalt in eine Datei auf dem Dateisystem.
t3lib_div::writeFile($file,$content)
// erzeugt eine Ausgabe der gewünschten Variablen zur Fehlersuche.
t3lib_div::debug($var="",$brOrHeader=0)
//lädt das $TCA für eine bestimmte Tabelle, um beispielsweise danach Änderungen
daran vorzunehmen.
t3lib_div::loadTCA($table)
// gibt den korrekten Klassennamen für eine zu initialisierende Klasse zurück.
t3lib_div::makeInstanceClassName($className)
// überprüft eine E-Mail-Adresse auf korrekte Syntax.
t3lib_div::validEmail($email)
// liest die Datei aus einer URL aus und gibt den Inhalt zurück.
t3lib_div::getURL($url, $includeHeader=0)
```

Listing 2

class t3lib_extMgm (class.t3lib_extmgm.php)

Dies ist die Hauptklasse für den Extension Manager, die ebenso wie „t3lib_div" nicht instanziiert wird, sondern deren Funktionen statisch aufgerufen werden. Die Klasse kommt in so gut wie jeder Extension irgendwie zum Einsatz. Beim Einbinden von Frontend-Plugins wird zum Beispiel schon vom Kickstarter folgende Zeile in der Datei „ext_tables.php" ergänzt:

PHP

```
t3lib_extMgm::addPlugin(array('LLL:EXT:abz_references/
locallang_db.xml:tt_content.list_type_pi1', $_EXTKEY.'_pi1'),'list_type');
```

Listing 3

Folgende Funktionen sollten Sie sich außerdem auf jeden Fall vormerken:

extPath($key,$script='')
Die Funktion „extPath" dient zur Erkennung des korrekten absoluten Pfads beim Einbinden einer Klasse aus der Extension „$key".

PHP

```
include_once(t3lib_extMgm::extPath('tt_news').'class.tx_ttnews_catmenu.php');
```

Listing 4

addPiFlexFormValue($piKeyToMatch,$value)
Die Funktion „addPiFlexFormValue" benötigen Sie beim Einsatz von Flexforms zur Konfiguration Ihres Plugins.

class t3lib_BEfunc (class.t3lib_befunc.php)

In dieser Klasse sind Funktionalitäten für das TYPO3-Backend gesammelt. Auch hier werden die Funktionen statisch, also ohne Instanziierung aufgerufen. Häufig eingesetzt werden folgende Funktionen:

deleteClause($table,$tableAlias='')
Diese Funktion sollten Sie in jede SQL-Abfrage im Backend einbauen, sobald die zugrunde liegende Tabelle über das $TCA konfiguriert ist, um über TYPO3 gelöschte Datensätze sauber zu behandeln.
getRecord($table,$uid,$fields='*',$where='')
Dies ist die ideale Funktion, um schnell und direkt einen Datensatz aus einer Tabelle zu holen, zu dem die uid bekannt ist.
getPagesTSconfig($id,$rootLine='',$returnPartArray=0)
Über das Page TSconfig Array können Sie Konfigurationsparameter für Ihre Module anbieten und auswerten.

class class t3lib_DB (class.t3lib_db.php)

Die Klasse „t3lib_DB" ist die generelle Wrapper-Klasse für jeglichen Datenbankzugriff. Allein betrachtet, stellt sie noch keine komplette Abstraktionsschicht dar, schafft aber die notwendige Grundlage dafür. Solange TYPO3 mit MySQL betrieben wird, werden die Zugriffsbefehle direkt in native MySQL-Befehle umgewandelt. Bei der Zusammenarbeit mit anderen Datenbanken kommt die Extension „dbal" [4] ins Spiel, die die jeweiligen Zugriffe steuert. Das Datenbank-Objekt steht in jedem TYPO3-Script zur Verfügung.

Klassische Nutzung von TYPO3_DB

```
$table = 'pages';
$where = $this->cObj->enableFields($table);
$groupBy = '';
$orderBy = 'sorting';
$limit = '';
$res = $GLOBALS['TYPO3_DB']->exec_SELECTquery (
  '*',
  $table,
  $where,
  $groupBy,
  $orderBy,
  $limit
);
$rows = array();
while ($row = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($res)) {
  $rows[] = $row;
}
$GLOBALS['TYPO3_DB']->sql_free_result($res);
```

Listing 5

Sie sollten möglichst nur Methoden, die mit „exec_*" beginnen einsetzen, da diese eine einwandfreie Zusammenarbeit mit der Extension „dbal" und so auch mit anderen Datenbanken ermöglichen.

class template (template.php)

Diese Klasse ist die zentrale Stelle für alle layout- und ausgabebezogenen Bereiche im Backend. Für einfache Backend-Module verrichtet sie ihren Dienst im Hintergrund und wird vom Programmierer (in vom Kickstarter erzeugten Extensions) gar nicht mehr bewusst eingesetzt.

class language (lang.php)

Mit Hilfe der Klasse „language" wird die gesamte Lokalisierungssteuerung von TYPO3 abgewickelt.

class tslib_pibase (tslib/class.tslib_pibase.php)

Diese Basisklasse für Frontend-Plugins stellt dem Entwickler ein grundlegendes Framework für den Einsatz im Frontend zur Verfügung. Fast alle aktuellen Plugins (mit Ausnahme der neuen MVC-basierten Plugins) sind eine abgeleitete Klasse von „tslib_pibase". Hier lohnt es sich ganz besonders, vor dem Erstellen des nächsten Plugins die von „tslib_pibase" zur Verfügung gestellten Möglich-

keiten zu studieren. Dies sind unter anderem Funktionen zum Erzeugen von Listen, zur Lokalisierung, zur Verlinkung sowie Blätter- und Suchfunktionen. Bei Bedarf können diese Funktionen auch überschrieben werden. Besonders interessante Methoden sind:

pi_linkToPage($str,$id,$target='',$urlParameters=array())
Die am häufigsten verwendete Funktion liefert einen kompletten HTML-Link (...) zu einer Seite innerhalb von TYPO3 zurück.
pi_linkTP($str,$urlParameters=array(),$cache=0,$altPageId=0)
Für Links auf dieselbe Seite mit zusätzlichen Parametern. Mithilfe des Parameters „$altPageId" können Sie eine andere Seite als die aktuelle Seite als Zielseite angeben.
pi_linkTP_keepPIvars($str,$overrulePIvars=array(),$cache=0, $clearAnyway=0,$altPageId=0)
Zusätzlich zu den Möglichkeiten von „pi_linkTP" werden alle Parameter des Plugins wieder in den Link integriert.
pi_list_browseresults($showResultCount=1,$tableParams='', $wrapArr=array(), $pointerName = 'pointer', $hscText = TRUE)
Sie können sehr komfortabel eine Blätterfunktion in Ihre Listendarstellung im Frontend integrieren. Vergleichen Sie die Darstellung mit der moderner Frontend-Plugins wie „ve_guestbook" oder „tt_news".
pi_list_searchBox($tableParams='')
Sie erhalten eine Suche basierend auf den Datensätzen des Plugins, die sich nahtlos in das Frontend-Framework einpasst.
pi_wrapInBaseClass($str)
Umschließen Sie jede Ausgabe aus Ihrem Plugin mit den entsprechenden -Tags. Dadurch werden Kollisionen von CSS-Angaben mit Ergebnissen anderer Plugins praktisch ausgeschlossen. Der Kickstarter schlägt die Verwendung von „pi_wrapInBaseClass" bereits richtig vor.
pi_getEditIcon($content,$fields,$title='',$row='',$tablename='', $oConf=array())
pi_getEditPanel($row='',$tablename='',$label='',$conf=Array())
Verwenden Sie diese Funktionen, um ein Frontend-Editing für die Datensätze Ihres Plugins zu ermöglichen.

class tslib_cObj (tslib/class.tslib_content.php)

Die umfangreichste Klasse im TYPO3-Framework ist zuständig für die Ausführung und Aufarbeitung der gesamten TypoScript-Angaben und stellt somit das Rückgrat der Seitengenerierung für das Frontend dar. Alle regulären TypoScript-Objekte sind in dieser Klasse enthalten.

PHP

```
function COBJ_ARRAY($conf,$ext='') {
  $content='';
  switch($ext) {
    case 'INT':
      $substKey = $ext.'_SCRIPT.'.$GLOBALS['TSFE']->uniqueHash();
      $content.='<!--'.$substKey.'-->';
      $GLOBALS['TSFE']->config[$ext.'incScript'][$substKey] = array(
        'file'=>$incFile,
        'conf'=>$conf,
        'cObj'=>serialize($this),
        'type'=>'COA'
      );
    break;
    default:
      if ($this->checkIf($conf['if.'])) {
        $content=$this->cObjGet($conf);
        if ($conf['wrap']) {
          $content=$this->wrap($content, $conf['wrap']);
        }
        if ($conf['stdWrap.']) {
          $content=$this->stdWrap($content, $conf['stdWrap.']);
        }
      }
    break;
  }
  return $content;
}
```

Listing 6

Den Zusammenhang zwischen TypoScript und PHP-Code sehen Sie sehr gut im Bereich „default" der switch-Anweisung. Dort wird geprüft, ob die einzelnen Konfigurationsmöglichkeiten in TypoScript wie „stdWrap" gesetzt wurden und die entsprechende Funktionalität aufgerufen wurde. Um beispielsweise ein Bild in einer Liste darzustellen, können Sie den Bildnamen aus der Datenbank lesen, alle weiteren Informationen (z. B. die Breite) in TypoScript hinterlegen und dann zur Erzeugung des img-Tags inklusive eines automatisch kleingerechneten temporären Bilds auf bestehende Funktionen zurückgreifen.

TypoScript

```
plugin.tx_myPlugin_pi1 {
  listView {
    myImage = IMAGE
    myImage {
      file.maxW = 50
    }
  }
}
```

Listing 7

PHP (Bild-Element erzeugen)

```
$conf['listView.']['myImage.']['file'] =
  'uploads/tx_myplugin/'.$this->internal['currentRow']['image'];
$content.= $this->cObj->IMAGE($conf['listView.']['myImage.']);
```

Listing 8

Dabei werden der Pfad und der Name der Bilddatei erst in PHP zum Konfigurationsarray hinzugefügt, um aus dem gesamten Konfigurationsarray „$conf['listView.']['myImage.']" dann das Bild und das img-Tag zu erzeugen. In der Variable „$this->internal ['currentRow']['image']" ist im Beispiel der Name der Datei aus der Datenbank hinterlegt.

class tslib_fe (tslib/tslib_fe.php)

Die Klasse enthält eine ganze Reihe von Funktionen und Attributen, die von der Hauptdatei im Frontend „index_ts.php" eingesetzt werden. Für den Extension-Entwickler bietet sie viele Möglichkeiten, um den aktuellen Zustand des Frontends abzufragen. Über die global verfügbare Variable „$GLOBALS["TSFE"]" können Sie einfach darauf zugreifen.

$GLOBALS["TSFE"]->id
Sie erhalten die uid der aktuellen Seite zurück.
$GLOBALS["TSFE"]->page
Der komplette Datensatz der aktuellen Seite steht Ihnen hiermit zur Verfügung.
$GLOBALS["TSFE"]->loginUser
Mit Hilfe von „$GLOBALS["TSFE"]->loginUser" prüfen Sie, ob der Betrachter der Seite sich als Frontend-Benutzer, beispielsweise für einen geschützten Bereich, angemeldet hat.
$GLOBALS["TSFE"]->fe_user->user
Per „$GLOBALS["TSFE"]->fe_user->user" greifen Sie anschließend auf den Datensatz des Benutzers zu.

Falls Sie innerhalb Ihres PHP-Codes auf eine beliebige TypoScript-Einstellung zugreifen müssen, können Sie die entsprechende Angabe analog zu den beiden Beispielen nutzen.

PHP

```
$GLOBALS['TSFE']->tmpl->setup['config.']['sys_language_uid'],
$GLOBALS['TSFE']->tmpl->setup['plugin.']['tx_testext_pi1.']['param']
```

Listing 9

TypoScript

```
config.sys_language_uid = 1
plugin.tx_testext_pi1.param = 3
```

Listing 10

Beachten Sie den Punkt am Ende der Bereichsbezeichnung, beispielsweise in ['config.']. Falls Sie auf Einstellungen der nächst tieferen Ebene zugreifen wollen, muss am Ende der Punkt stehen – und zwar so oft, bis das endgültige Zielelement erreicht ist.

Fazit

Die Serie zur Entwicklung von TYPO3-Extensions hat Ihnen alle wichtigen Einstiegspukte sowie Tipps zu weiterführenden Infos vermittelt. Viel Spaß und Erfolg mit Ihren eigenen Extensions.

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 2059

[1] Extension cc_beinfo: http://typo3.org/extensions/repository/view/cc_beinfo/1.0.0/

[2] Dokumentation der TYPO-Core-API: http://typo3.org/documentation/document-library/core-documentation/doc_core_api/current/view/

[3] Extension extdeveval: http://typo3.org/extensions/repository/view/extdeveval/3.0.0/

[4] Extension dbal: http://typo3.org/extensions/repository/view/dbal/0.9.9/

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.