

TYPO3 Extension Entwicklung

21TORR AGENCY GmbH | Schritt-für-Schritt Anleitung zum Entwickeln von TYPO3 Extensions

Version, Status: 1.0, final

Datum: 20. Mai 2009

21TORR AGENCY gmbh

Heinestrasse 72

72762 Reutlingen

Autor: Kai Vogel

Telefon: 49-7121-348-217 E-Mail: k.vogel@21torr.com

Dieses Dokument darf – mit Nennung des Autoren – frei vervielfältigt, verändert und weitergegeben werden.

Der Inhalt ist sorgfältig recherchiert, mit dem Dokument ist jedoch keinerlei Garantie auf Fehlerfreiheit gewährleistet.

Dieser Inhalt ist unter einem Creative Commons Lizenzvertrag lizenziert. Um die Lizenz anzusehen, gehen Sie bitte zu <http://creativecommons.org/licenses/by/3.0/de/> oder schicken Sie einen Brief an Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

Inhaltsverzeichnis

1 Überblick.....	3
2 Vorüberlegungen.....	3
3 Kickstarter.....	4
4 Die erste eigene Extension.....	4
4.1 Mehrsprachigkeit.....	5
4.2 Eine neue Tabelle.....	5
4.3 Das Frontend Plugin.....	6
4.4 Installation.....	7
5 Was kommt dann?.....	8
6 Wie unterstützt mich TYPO3?.....	10
6.1 Das Content Element.....	11
6.2 Datenbankabstraktion.....	12
6.3 Links.....	13
6.4 Mehrsprachigkeit.....	14
6.5 Konfiguration.....	15
6.6 Templates.....	16
6.7 XCLASS.....	21
7 Was ist der TCA?.....	21
8 Dateien im Detail.....	23
8.1 ext_emconf.php.....	23
8.2 ext_localconf.php.....	23
8.3 ext_tables.php.....	23
8.4 ext_tables.sql.....	23
8.5 tca.php.....	24
8.6 pi1/class.tx_<extension-key>_pi1.php.....	24
8.7 pi1/class.tx_<extension-key>_pi1_wizicon.php.....	24
9 Weiterführende Informationen.....	24

1 Überblick

Die größte Stärke von TYPO3 ist seine Modularität. Durch sogenannte Extensions [5] ist es möglich, die Funktionsweise des Kernsystems [7] zu erweitern. Diese Extensions findet man hauptsächlich in einer Online-Datenbank – dem TER (TYPO3 Extension Repository). Hier kann man jede beliebige Extension kostenlos herunterladen. Erstellt und gepflegt werden diese überwiegend durch die Community. [6]

Da TYPO3 selbst für große Internetauftritte immer attraktiver wird, werden auch die Anforderungen an die Extensions immer vielfältiger. Und da es nicht für jeden Zweck eine passende Erweiterung gibt, bietet das vorliegende Dokument eine Schritt-für-Schritt Anleitung zur Erstellung einer eigenen Extension. [1]

Nachfolgend werden wir ein Frontend-Plugin erstellen, welches durch eine Listen- und Detailansicht Datensätze aus der Datenbank im Frontend anzeigt und die TYPO3-üblichen Formulare zu deren Pflege im Backend bereitstellt.

2 Vorüberlegungen

Bevor wir uns an den praktischen Teil wagen, sollten wir zuerst einmal die Problemstellung identifizieren. Nehmen wir also an, eine Anforderung sei es, die zahlreichen Online-Kommunikationsdienste auf einer Seite des Internetauftritts aufzulisten. Das lässt sich einfach über einen Text-Datensatz im Backend von TYPO3 realisieren. Doch was, wenn diese Liste dynamisch sein soll? Eine weitere Anforderung ist es zudem, für jeden Dienst eine Detailansicht anzubieten. Jetzt kann man zwar für jeden Kommunikationsdienst der Liste eine eigene Seite anlegen, jedoch erhöht sich dadurch der Verwaltungsaufwand um ein Vielfaches.

Zum jetzigen Zeitpunkt lässt sich schon recht genau sagen, welche Anforderungen unsere Extension erfüllen soll:

- Es soll eine Listen- und Detailansicht für das Frontend bereit gestellt werden

- Die Kommunikationsdienste sollen dynamisch pflegbar sein
- Der Verwaltungsaufwand soll so gering wie möglich gehalten werden
- Die Datensätze sollen vom TYPO3-üblichen Komfort zur Bearbeitung Gebrauch machen

Wir werden im Rahmen dieser Anleitung genau diese Punkte umsetzen und dafür die TYPO3 spezifischen Methoden verwenden.

Eine davon ist der Extension Kickstarter. Es handelt sich dabei um eine Extension (kickstarter), welche wie alle anderen Erweiterungen über den Extension-Manager installiert werden muss.

3 Kickstarter



Abbildung 1: Kickstarter

Einmal installiert wird der Kickstarter als weiterer Punkt in der Auswahlliste des Extension-Managers hinzugefügt. Wählen Sie dort „Make new extension“ um das neue Modul zu öffnen. Es präsentiert sich, wie in Abbildung 1 zu sehen, auf den ersten Blick eine eher überschaubare Menge an Optionen. Lediglich ein Feld für den Extension-Key und einige Stichpunkte mit anliegendem Plus-Zeichen sind zu sehen.

Widmen wir uns zuerst dem Eingabefeld. Hier muss für jede neue Extension ein eindeutiger Name, der sogenannte Extension-Key definiert werden. Dieser kann bei einer lokalen Extension beliebig gewählt werden. Sollte sie jedoch veröffentlicht werden muss über den Link unter dem Eingabefeld ein Key registriert werden.

Wir wählen den Namen „**imtools**“ und beginnen nun mit der Konfiguration unserer Extension.

4 Die erste eigene Extension

Zuerst klicken Sie auf das Plus-Zeichen neben „General info“. Im nun angezeigten Formular vergeben Sie einen Titel und eine Beschreibung. Als Kategorie eignet sich „Frontend Plugin“, da es sich bei unserer Extension um eine Frontendausgabe handelt. Anschließend befüllen Sie noch die Felder für Autor und E-Mail Adresse.

4.1 Mehrsprachigkeit

TYPO3 unterstützt momentan bereits eine Vielzahl Sprachen, davon sollten Sie Gebrauch machen. Klicken Sie dafür auf das Plus neben „Setup languages“ um das nächste Formular zu öffnen. Wählen Sie nun „German“ um die neue Extension nicht nur in Englisch, sondern auch in Deutsch anbieten zu können. Die verschiedensprachigen Versionen der Label werden innerhalb der Extensions in Form von XML-Dateien abgelegt.

4.2 Eine neue Tabelle

Öffnen Sie nun das Formular für eine neue Tabelle „New Database Tables“. In dieser Tabelle werden später die Kommunikationsdienste in Form von Datensätzen gespeichert. Die Tabelle bekommt einen Namen und einen Titel in Englisch. Zusätzlich können Sie nun einen Titel in den Sprachen angeben, die im vorherigen Schritt ausgewählt wurden.

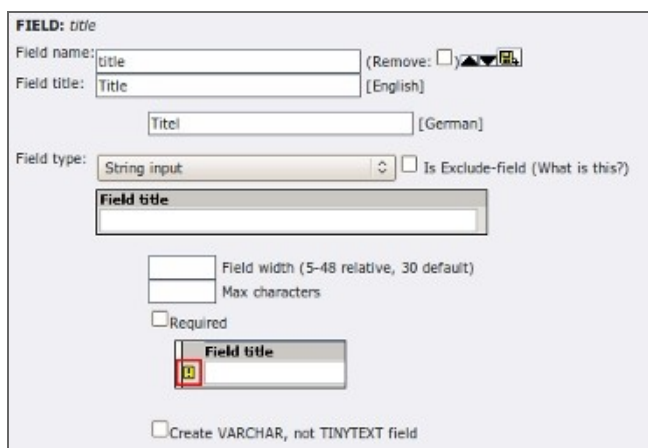
Betrachten wir nun die weiteren Optionen. Hier können Sie angeben, ob Datensätze nur als gelöscht markiert, oder physikalisch aus der Datenbank entfernt werden sollen und ob sie versteckt, lokalisiert oder versioniert werden können. Außerdem ist es möglich, weitere Felder zum zeitgesteuerten Anzeigen der Datensätze hinzuzufügen.

Wenn die Auflistung im Frontend sortierbar sein soll, wählen Sie die Option „Manual ordering of records“. Alle weiteren Konfigurationsmöglichkeiten lassen wir jetzt erstmal außer Acht und gehen direkt über zur Definition der Tabellen-Felder am Ende des Formulars.

Hier können Sie Tabellen-Felder anlegen. Es werden dabei automatisch die Eingabe- und Auswahlfelder des Formulars für die Datensatzpflege im Backend und die entsprechende Datenbanktabelle mit allen dafür benötigten Spalten erzeugt.

Legen Sie zuerst das Feld „Titel“ an, um die Kommunikationsdienste später benennen zu können. Dafür vergeben Sie den Feldnamen „title“ und einen Titel in den verschiedenen Sprachen. Da es sich bei dem Wert des Feldes um einen Text handeln wird, empfiehlt sich an dieser Stelle der Feldtyp „String input“.

Nach einem Klick auf „Update“ werden zusätzliche Attribute des Eingabefelds geöffnet, dies wird in Abbildung 2 dargestellt. Schauen wir sie uns im Detail an. Die Feldbreite und Anzahl Zeichen entscheiden über die Darstellung des Eingabefeldes und mit „Required“ kann festgelegt werden, ob das Eingabefeld ausgefüllt werden muss, damit der Datensatz gespeichert werden kann.



The screenshot shows a configuration window for a field named 'title'. It includes input fields for the field name, title in English ('Title'), and title in German ('Titel'). The field type is set to 'String input'. Below this, there are options for 'Field width (5-48 relative, 30 default)', 'Max characters', and a 'Required' checkbox. A preview of the field shows a text input with a red border and a warning icon. At the bottom, there is a checkbox for 'Create VARCHAR, not TINYTEXT field'.

Abbildung 2: Zusätzliche Attribute

Auf die gleiche Weise erzeugen Sie nun noch weitere Felder für die Bezugsadresse, Lizenz, Version und weitere Informationen. Zu beachten ist allerdings, dass keiner der darüber gelisteten reservierten Feldnamen ein zweites Mal vergeben werden darf, da es sonst zu Konflikten kommt.

Im Anschluss vergeben Sie weiter oben im Formular noch ein „Label-field“, wählen Sie dafür das eben erzeugte Feld „title“.

4.3 Das Frontend Plugin

Nachdem nun die Tabelle für die Kommunikationsdienste vorbereitet ist, erstellen wir jetzt einen Datensatz, der diese im Frontend, in Form einer Listen- und Detailansicht, ausgibt. Dazu öffnen Sie das Formular für ein neues Frontend Plugin. Hier vergeben Sie einen Titel und wählen aus den Einbindungsmöglichkeiten für das Plugin den Eintrag in der „Insert plugin“-Liste. Zusätzlich können Sie, wie in Abbildung 3 zu sehen, ein eigenes Symbol im „Neuer Datensatz“-Assistent hinzufügen. Anschließend bestätigen Sie die Auswahl durch einen Klick auf „Update“ am Ende des Formulars.



Abbildung 3: „Neuer Datensatz“-Assistent

4.4 Installation

Damit ist die Grundstruktur unserer Erweiterung fertig. Betrachten Sie nun das Ergebnis, indem Sie auf Button „View result“ klicken. Unterhalb der nun dargestellten Dateiliste werden die wichtigsten davon bereits angezeigt, wir schauen sie uns jedoch erst später im Detail an.

Schreiben Sie nun die Extension in das Dateisystem indem Sie auf „Write“ klicken. Es erscheint eine Meldung, die Sie genau lesen sollten.

Der Kickstarter ist kein Editor, sondern dient rein dem Anlegen der Grundstruktur!

Dieser Hinweis ist wichtig, denn häufig passiert es, dass Einsteiger die eigene Extension noch mal mit dem Kickstarter bearbeiten und ärgern sich anschließend darüber, dass ihre Änderungen in den Dateien dadurch überschrieben wurden.

Wenn Sie nun auf „Install extension“ klicken, wird die neue Erweiterung ganz normal über den Extension-Manager installiert. Wie in Abbildung 4 dargestellt, wird im nächsten Dialog noch das SQL-Statement angezeigt, welches die neue Datenbanktabelle erstellt. Führen Sie dieses Update aus und betrachten Sie den neuen Eintrag in der Extensionliste.

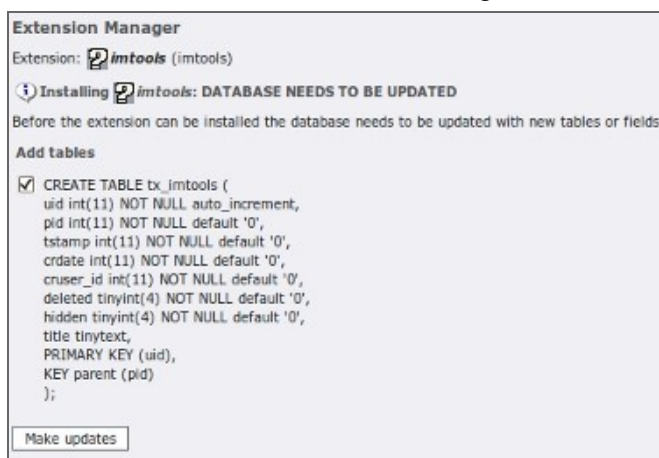


Abbildung 4: SQL-Statement

5 Was kommt dann?

Nachdem die Extension nun vorbereitet wurde, können wir dazu übergehen, die Frontendausgabe anzupassen. Jedoch sollten Sie vorher einen neuen Sysfolder im Seitenbaum und darin einige Kommunikationsdienste anlegen. Anschließend benötigten Sie noch auf einer beliebigen Seite einen Frontend-Datensatz der neuen Extension. Dieser wird zum Testen der Änderungen am PHP-Code unseres Plugins verwendet.


```

);
$rows = array();
while($row = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($resource)) {
    // Add header only one times
    if (!count($rows)) {
        $rows[] = '<tr><th>'.implode('</th><th>', array_keys($row)).
            '</th><th>&nbsp;</th></tr>';
    }
    // Add rows
    $link = $this->cObj->typolink('Detail', array(
        'parameter' => $GLOBALS['TSFE']->id,
        'additionalParams' => '&single_uid='.(int) $row['uid'],
    ));
    $rows[] = '<tr><td>'.implode('</td><td>', array_values($row)).
        '</td><td>'.$link.'</td></tr>';
}
$content = '<table>'.implode('', $rows).'</table>';
} else {
    // Singleview
    $resource = $GLOBALS['TYPO3_DB']->exec_SELECTquery(
        '*', // SELECT
        'tx_imtools', // FROM
        'uid='.(int)$parameter.$exclude, // WHERE
        '', // GROUP BY
        '', // ORDER BY
        '1' // LIMIT
    );
    $result = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($resource);
    foreach ($result as $key => $value) {
        $content .= '<tr><th>'.$key.':</th><td>'.$value.'</td></tr>';
    }
    $link = $this->pi_linkToPage('Back', $GLOBALS['TSFE']->id);
    $content = '<table>'.$content.</table><p>'.$link.</p>';
}
return $this->pi_wrapInBaseClass($content);
}

```

Die hier eingesetzten Methoden werden in Kapitel 6 im Detail beschrieben.

Wenn Sie nun die Datei speichern und Ihre Seite im Frontend betrachten, werden Sie feststellen, dass bereits beide Ansichten verfügbar sind.

6 Wie unterstützt mich TYPO3?

TYPO3 bietet eine ganze Reihe an Methoden, auf die Entwickler zurückgreifen können. Einen Überblick darüber findet man in der Core-API-Reference. [8] Es wird empfohlen wenn möglich immer auf die TYPO3 eigenen Methoden zurückzugreifen, statt mit den gewohnten Funktionen zu arbeiten. Im Folgenden werden Sie einige davon kennen lernen.

Nur durch die TYPO3-Methoden kann gewährleistet werden, dass das gesamte System mit allen Extensions einheitlich aufgebaut ist. Durch eigene Funktionen entstehen zum Beispiel Schwachstellen in der Sicherheit des Systems. Außerdem wird eine Änderung an einer TYPO3-Methode automatisch in allen Extensions übernommen, so können Bugs schnell behoben werden.

Zudem sollte bei der Extension-Entwicklung der vorgegebene Programmierstil eingehalten werden. [4]

6.1 Das Content Element

Jede Extension wird von der Klasse „tslib_pibase“ aus der Datei „.../typo3_src/typo3/sysext/cms/tslib/class.tslib_pibase.php“ abgeleitet. Somit hat auch jede Extension Zugriff auf die darin enthaltenen Methoden. Das ist quasi die Grundausstattung eines Extension-Entwicklers.

Eine der Methoden ist folgende:

```
$this->pi_wrapInBaseClass();
```

Sie sorgt dafür, dass die gesamte Ausgabe der neuen Extension mit einem DIV umschlossen wird. So ist eine absolute Unterscheidbarkeit der einzelnen Extensions im DOM (Dokument Object Model) gewährleistet. Das macht es zum Beispiel einfacher, nur die Ausgabe unserer Extension getrennt vom eigentlichen Inhalt per CSS (Cascading Style Sheets) zu stylen.

Außerdem hat man innerhalb der Plugin-Klasse überall Zugriff auf die Variable „cObj“, welche eine Instanz der Klasse „tslib_cObj“ darstellt. Diese Instanz wird automatisch beim Laden der Extension zugewiesen und ermöglicht den Zugriff auf dessen Methoden. So können Sie zum Beispiel mit folgendem Aufruf einen Text ab einer Länge von 300 Zeichen abschneiden:

```
$this->cObj->crop($content, '300 | ... | 1');
```

6.2 Datenbankabstraktion

TYPO3 geht einen eigenen Weg um mit Datenbanktabellen zu kommunizieren. Alle Datenbank-Zugriffe werden durch eigene Methoden vom normalen SQL-Syntax abstrahiert. Dadurch kann sichergestellt werden, dass die Installation unter verschiedenen Datenbanksystemen (MySQL, Oracle, Postgres, ...) identisch funktioniert. Außerdem wird auch hier, wie überall in TYPO3, der Fokus auf die Sicherheit und Erweiterbarkeit des Systems und des Internetauftritts gelegt. Denn bei eigenen SQL-Statements und Datenbankverbindungen kann nicht sichergestellt werden, dass diese konsistent vor SQL-Injection schützen.

Zum Beispiel werden mit den TYPO3-Methoden Datensätze heraus gefiltert, die im Backend als versteckt markiert wurden, oder deren Anzeigezyklus abgelaufen ist. Des Weiteren werden Datensätze von TYPO3 selten wirklich gelöscht, sondern nur als gelöscht markiert. Dadurch ist es möglich diese über eine Historie wiederherstellen. Auch diese Datensätze werden aus der Ergebnismenge entfernt. Dies wird erreicht, indem man die folgende Methode zur „WHERE“-Bedingung des Statements hinzufügt:

```
$exclude = $this->cObj->enableFields('table_name');
```

Es werden für jeden erdenklichen Fall Methoden zur Verfügung gestellt. So wird es dem Entwickler leicht gemacht, Datensätze auszulesen, anzulegen, oder zu löschen. Hier ein Beispiel eines SELECT-Statements:

```
$exclude = $this->cObj->enableFields('pages');
$result = $GLOBALS['TYPO3_DB']->exec_SELECTquery(
    '*',          // SELECT
    'pages',      // FROM
    'pid=1'.$exclude, // WHERE
    '',          // GROUP BY
    'uid',       // ORDER BY
    '10'        // LIMIT
);
```

Grundsätzlich werden in TYPO3 Beziehungen von Datensätzen zu anderen Tabellen über Komma-Separierte Listen in einem Tabellenfeld realisiert. Erst seit Kurzem werden echte m-zu-m-Beziehungen unterstützt. Dazu wurden sogar eigens neue Methoden erstellt. Allerdings wird diese Technik von Extensions bisher nur vereinzelt wirklich eingesetzt.

Für Referenzierungen untereinander gibt es seit der Version 4.0 eine neue Tabelle. So kann man direkt an einem Datensatz sehen, wie viele und welche Elemente der Website auf diesen referenzieren. Nehmen wir zum Beispiel eine Seite die von mehreren Listenansichten als gemeinsame Detailseite genutzt wird. Dank der neuen Tabelle kann man herausfinden, wie viele Listenansichten die aktuelle Seite als Detailansicht nutzen. Außerdem wird dadurch eine Meldung ausgegeben, wenn man einen referenzierten Datensatz löschen möchte.

6.3 Links

Links werden in TYPO3 durch die TypoLink-Methode erstellt. Dies hat den Vorteil, dass alle Links identisch behandelt werden. So ist es zum Beispiel möglich, auf einen Schlag allen Links ein zusätzliches Attribut hinzuzufügen, egal, ob die Links im TypoScript, [3] oder durch eine Extension erstellt wurden.

Die Methode erwartet mindestens das Ziel des Links und erzeugt daraus bereits einen vollwertigen Link, mit „title“ und, je nach Ziel, einem passenden „target“. Bei dem Ziel darf es sich natürlich auch um die ID einer Seite im Seitenbaum, eine E-Mail Adresse oder eine Datei im Dateisystem handeln.

Gibt man der Methode als weitere Anweisung den Befehl, dass sie die vorhergehende URL zurück geben soll, so erhält man nur die URL ohne A-Tag.

Ein Aufruf der TypoLink-Methode gestaltet sich wie folgt:

```
$myLink = $this->cObj->typoLink('Link Text', array(
    'parameter' => 'google.de',
));
```

Als Ergebnis wird folgender Link geliefert:

```
<a href="http://www.google.de" title="" target="_blank">Link Text</a>
```

Vereinfacht kann man einen Link auf interne Seiten auch mit den folgenden Methoden erstellen:

```
// http://www.domain.com/index.php?id=5
$myLink = $this->pi_getPageLink(5);

// <a href="http://www.domain.com/index.php?id=5" target="_self">Link Text</a>
$myLink = $this->pi_linkToPage('Link Text', 5);
```

6.4 Mehrsprachigkeit

Wie bereits zu Beginn des Dokuments im Kapitel 4.1 beschrieben, unterstützt TYPO3 eine ganze Reihe verschiedener Sprachen. Die Label jeder einzelnen Sprache werden in XML-Dateien hinterlegt. Darin enthalten ist immer ein Standard-Fallback und für jede Sprache ein eigener Bereich. Der Kickstarter legt für unsere Extension drei Sprachdateien an:

- locallang.xml

- locallang_db.xml
- pi1/locallang.xml

Letztere enthält alle Label, die unser Frontend-Plugin für die Darstellung benötigt. Die beiden anderen enthaltenen hingegen alle, im Backend relevanten Label. Generell ist es nicht notwendig, drei oder mehr Sprachdateien zu nutzen, so kann man dies auf zwei Dateien reduzieren, eine für das Frontend und eine für das Backend. Jedoch ist darauf zu achten, dass diese Dateien richtig eingebunden werden.

Um im Frontend-Plugin mehrsprachige Label nutzen zu können, müssen sie erst aus der XML-Datei in einen Array geladen werden. Dies geschieht über einen Funktionsaufruf am Anfang der Hauptmethode:

```
$this->pi_loadLL();
```

Anschließend kann man über den folgenden Aufruf auf Label zugreifen:

```
$header = '<h1>'.$this->pi_getLL('LABEL_NAME').'</h1>';
```

6.5 Konfiguration

Für unsere Extension ist es nun möglich, im TypoScript [3] eine Konfiguration zu erstellen. Diese lässt sich anschließend im PHP-Code auslesen und verarbeiten. Hierfür stehen eine Vielzahl Methoden zur Verfügung. Das TypoScript wird beim Aufruf unserer Hauptmethode als Parameter „\$conf“ übergeben. Folgender TypoScript...

```
plugin.tx_imtools_pi1 {
    markers {
        MARKER1 = TEXT
        MARKER1.value = Wert
    }
}
```

...wird in dieser Form in den Konfigurationsarray unserer Klasse geschrieben:

```
$this->conf = array (
    'markers.' => array (
        'MARKER1' => 'TEXT',
        'MARKER1.' => array (
            'value' => 'Wert',
        ),
    ),
)
```

Sie können jetzt folgendermaßen darauf zugreifen (achten Sie auf den Punkt):

```
$markerWert = $this->conf['markers.']['MARKER1.']['value'];
```

Um nun nicht jeden Wert einzeln auslesen zu müssen, lässt man sich das TypoScript-Objekt (TEXT) am Besten mit folgendem Aufruf interpretieren:

```
$markers = $this->conf['markers.'];
$markerWert = $this->cObj->cObjGetSingle($markers['MARKER1'], $markers['MARKER1.']);
```

6.6 Templates

Templates sind in TYPO3 HTML-Dateien. Diese werden durch sogenannte Subparts in einzelne Bereiche unterteilt. So kann in der gleichen Datei ein Template für eine Listen- und eine Detailansicht untergebracht werden. Durch entsprechende Methoden wird anschließend das Template in unseren Plugin-Code geladen:

```
$template = $this->cObj->fileResource('PFAD/ZUR/DATEI.html');
$listView = $this->cObj->getSubpart($template, '###LIST###');
$singleView = $this->cObj->getSubpart($template, '###SINGLE###');
```

Das passende Template sieht dazu wie folgt aus:

```
<html>
...
  <body>
    <!-- ###LIST### begin -->
    ...
    <!-- ###LIST### end -->
    <!-- ###SINGLE### begin -->
    ...
    <!-- ###SINGLE end -->
  </body>
</html>
```

Nachdem wir nun den HTML-Code für unsere Listen- und Detailansicht haben, geht es damit weiter, dass wir die Marker ersetzen. Marker sind einfache Platzhalter im Template. Diese werden durch folgende Methode ersetzt:

```
$markers = array(
    '###MARKER1###' => 'Wert1',
    '###MARKER2###' => 'Wert2',
);
// Replace markers
$content = $this->cObj->substituteMarkerArray($listView, $markers);
```

Das Listen-Template gefüllt mit den oben genannten Markern sieht dann wie folgt aus:

```
<!-- ###LIST### begin -->
  <div>
    <p>###MARKER1###</p>
    <p>###MARKER2###</p>
  </div>
<!-- ###LIST### end -->
```

Nehmen wir nun noch einmal unser Code-Beispiel aus Kapitel 5 zur Hand. Hier wird die gesamte Ausgabe im PHP-Code der Extension definiert. Was aber, wenn die Kommunikationsdienste nun nicht mehr in Form einer Tabelle, sondern in einer unsortierten Liste (UL) ausgegeben werden sollen? In diesem Fall müsste der PHP-Code angepasst werden. Das ist umständlich und zeitaufwändig.

Daher ändern wir im Folgenden die Hauptmethode um ein HTML-Template nutzen zu können. Legen Sie dafür ein neues Verzeichnis namens „res“ im Hauptverzeichnis der Extension an. Erstellen Sie dort eine Datei namens „template.html“ und fügen Sie folgenden Inhalt ein:

```
<!-- ###LIST### begin -->
  <table>
    <tr>
      <th>uid</th>
      <th>pid</th>
      <th>title</th>
      <th>&nbsp;</th>
    </tr>
    <!-- ###LIST_ITEM### -->
    <tr>
      <td>###UID###</td>
      <td>###PID###</td>
      <td>###TITLE###</td>
      <td>###LINK###</td>
    </tr>
    <!-- ###LIST_ITEM### -->
  </table>
<!-- ###LIST### end -->

<!-- ###SINGLE### begin -->
  <table>
    <tr>
      <th>uid:</th>
      <td>###UID###</td>
    </tr>
    <tr>
      <th>pid:</th>
```



```

$subparts = array(
    'content' => $this->cObj->getSubpart($template, '###LIST###'),
    'item'    => $this->cObj->getSubpart($template, '###LIST_ITEM###'),
);

$rows = array();
while($row = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($ressource)) {
    $markers = array();
    foreach($row as $key => $value) {
        $markers['###'.strtoupper($key).'###'] = $value;
    }
    $markers['###LINK###'] = $this->cObj->typolink('Detail', array(
        'parameter'          => $GLOBALS['TSFE']->id,
        'additionalParams' => '&single_uid='.(int) $row['uid'],
    ));

    $rows[] = $this->cObj->substituteMarkerArray(
        $subparts['item'],
        $markers
    );
}

$items = array('###LIST_ITEM###' => implode(PHP_EOL, $rows));
$content = $this->cObj->substituteMarkerArrayCached(
    $subparts['content'],
    array(),
    $items
);
} else {
    // Singleview
    $ressource = $GLOBALS['TYPO3_DB']->exec_SELECTquery(
        '*',                               // SELECT
        'tx_imtools',                       // FROM
        'uid='.(int) $parameter.$exclude,  // WHERE
        '',                                  // GROUP BY
        '',                                  // ORDER BY
        '1'                                  // LIMIT
    );
}

```

```

);

$result = $GLOBALS['TYPO3_DB']->sql_fetch_assoc($ressource);

$markers = array();
foreach ($result as $key => $value) {
    $markers['###'.strtoupper($key).'###'] = $value;
}

$markers['###BACK_LINK###'] = $this->pi_linkToPage(
    'Back',
    $GLOBALS['TSFE']->id
);

$subpart = $this->cObj->getSubpart($template, '###SINGLE###');
$content = $this->cObj->substituteMarkerArray($subpart, $markers);
}

return $this->pi_wrapInBaseClass($content);
}

```

Wenn Sie sich nun das Ergebnis im Frontend ansehen, werden Sie feststellen, dass die Ausgabe der vorherigen recht ähnlich sieht, allerdings auf ein paar wenige Spalten reduziert wurde. Erweitern wir unsere Datenbank-Tabelle nun um eine Spalte, so wird diese nicht direkt im Frontend dargestellt, sondern muss erst per Hand im Template definiert werden.

6.7 XCLASS

Jede Extension sollte eine Schnittstelle zum Überschreiben von vorhanden oder hinzufügen von neuen Methoden besitzen. Am Besten jede die gleiche. Daher sind in fast jeder Extension die folgenden Zeilen in der Plugin-Datei zu finden:

```

if (defined('TYPO3_MODE') && $TYPO3_CONF_VARS[TYPO3_MODE]['XCLASS']
['ext/imtools/pil/class.tx_imtools_pil.php']) {
    include_once($TYPO3_CONF_VARS[TYPO3_MODE]['XCLASS']

```

```
[ 'ext/imtools/pil/class.tx_imtools_pil.php' ] );  
}
```

Diese Zeilen erlauben die Anwendung der XCLASS-Methode und sollten gewissenhaft gepflegt werden. [5]

7 Was ist der TCA?

Beim TCA (Table Configuration Array) [2] handelt es sich um das Herzstück von TYPO3. Er enthält den gesamten Aufbau der Datenbanktabellen, ihrer Beziehungen untereinander und Informationen über die Darstellung der Tabellenfelder in Backend-Formularen. Fast das gesamte Backend ist mit diesem Array verbunden. Selbst die Art und Weise, wie Seiten und Datensätze im Backend angezeigt werden, ist im TCA hinterlegt.

Alle Extensions und deren Tabellenkonfiguration werden in diesen Array geladen und können entsprechend von anderen Extensions manipuliert werden. Dadurch ist eine hohe Flexibilität beim Erstellen von eigenen und beim Anpassen von anderen Extensions gegeben. Es besteht sogar die Möglichkeit, Änderungen direkt am Herzen von TYPO3 vorzunehmen, da TYPO3 selbst auch zum größten Teil durch Extensions im TCA vertreten ist. Diese Freiheit sollte von Entwicklern allerdings mit Bedacht genutzt werden.

Alle Anpassungen werden wieder aus dem TCA entladen, sobald man die entsprechende Erweiterung im Extension-Manager deaktiviert. Somit kann schnell Abhilfe geschaffen werden, wenn eine Fehlfunktion auftaucht. Sollte der Extension-Manager nicht mehr aufzurufen sein, muss die Erweiterung manuell aus der Extension-Liste in der localconf.php entfernt werden.

Für unsere Extension ist die komplette Konfiguration des TCA in den folgenden Dateien enthalten:

- tca.php
- ext_tables.php

Hier werden nicht nur die einzelnen Tabellenfelder und ihre Darstellung im Backend-Formular definiert, sondern auch, wie die Datensätze in der Listenansicht des Backends aufgelistet werden, oder welche Buttons dabei zur Verfügung stehen.

Des Weiteren ist es möglich, eine Dynamik in die Darstellung der Felder zu integrieren. Sollen zum Beispiel nicht immer alle Felder gleichzeitig im Formular angezeigt werden, sondern jeweils nur ein Teil, je nach dem, welche Auswahl in einem Auswahlfeld getroffen wurde, so kann dies mit Hilfe von sogenannten „Types“ umgesetzt werden. Anwendung finden die Types, wie in Abbildung 5 zu sehen, zum Beispiel in der Auswahl des Inhaltstyps:

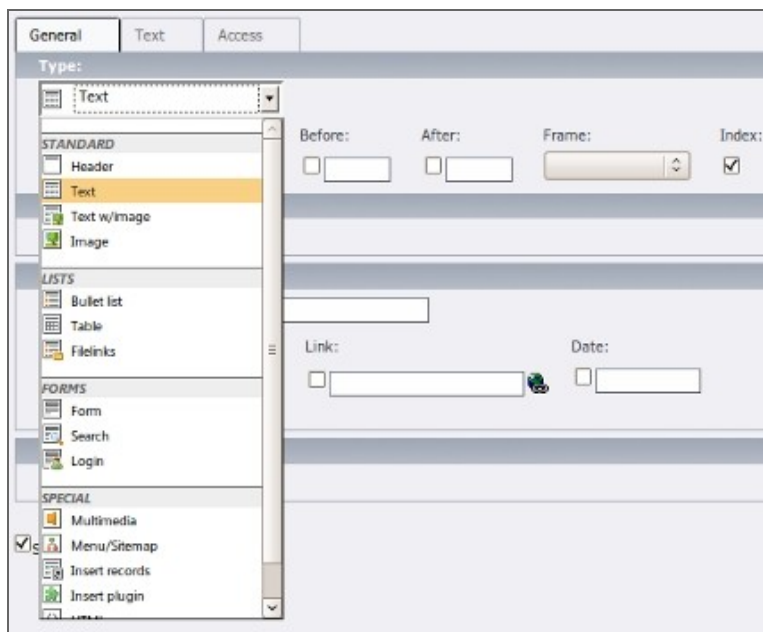


Abbildung 5: Types im Beispiel der Auswahl eines Inhaltstyps

Der TCA wird automatisch im TYPO3-Cache vorgehalten und nicht jedes Mal neu erstellt. Dadurch werden Änderungen nicht automatisch sichtbar. Abhilfe schafft ein Leeren des gesamten Caches.

8 Dateien im Detail

8.1 ext_emconf.php

Diese Datei enthält die Konfiguration für die Anzeige im Extension-Manager. Konfigurierbar sind hier unter Anderem Titel, Kategorie, Autor und Abhängigkeiten. Des Weiteren beinhaltet sie einen serialisierten Array mit einem MD5-Hash aller Extension-Dateien. Dieser erzeugt im Extension-Manger eine Liste der vom Original abweichenden Dateien. Das soll verhindern, dass die Erweiterungen unbemerkt verändert werden.

8.2 ext_localconf.php

Die ext_localconf.php erweitert die TYPO3 eigene localconf.php. Hier können Anpassungen der Extension-, System-, Backend- und Frontendkonfiguration vorgenommen sowie Hooks und XCLASS-Methoden definiert werden.

8.3 ext_tables.php

Mit Hilfe dieser Datei kann der globale Tabellenkonfigurations-Array (TCA) manipuliert werden. Hier wird das Backend-Formular der Extension mit den Spalten der Datenbanktabelle verknüpft. Außerdem ist es zum Beispiel möglich den Backend-Formularen anderer Extensions neue Eingabefelder hinzuzufügen.

8.4 ext_tables.sql

Die SQL-Statements in dieser Datei werden von TYPO3 interpretiert und in der Datenbank umgesetzt. Somit werden CREATE-Statements nicht blind ausgeführt, sondern ausgewertet und im Falle einer vorhandenen Tabelle mit dieser zusammengeführt (ALTER). Im Beispiel unserer

Extension wurde dadurch die neue Datenbanktabelle erstellt. Außerdem kann man so andere Tabellen erweitern. Nicht möglich ist es Tabellen oder Inhalte zu löschen oder neue Inhalte einzufügen.

8.5 tca.php

Die tca.php enthält die gesamte Konfiguration der Tabellenfelder [2] unserer Extension und wird in der Datei ext_tables.php definiert.

8.6 pi1/class.tx_<extension-key>_pi1.php

In dieser Datei ist die Hauptmethode unserer Extension enthalten. Diese wird aufgerufen, wenn das Plugin im Frontend angezeigt wird. Sie erzeugt die gesamte Ausgabe und definiert, ob unser Plugin gecached werden soll, oder nicht.

8.7 pi1/class.tx_<extension-key>_pi1_wizicon.php

Hier wird unsere Extension der „Insert plugin“-Liste und dem „Neuer Datensatz“-Assistenten hinzugefügt. Außerdem wird für diesen Zweck ein Symbol definiert.

9 Weiterführende Informationen

[1] Extension-Entwicklung :

<http://www.addison-wesley.de/main/main.asp?page=deutsch/bookdetails&ProductID=108830>

[2] TCA:

http://typo3.org/documentation/document-library/core-documentation/doc_core_api/4.0.0/view/4/1/

[3] TypoScript Reference:

http://typo3.org/documentation/document-library/core-documentation/doc_core_tsref/4.2.0/view/

[4] Coding-Guidelines:

http://typo3.org/documentation/document-library/core-documentation/doc_core_cgl/4.1.0/view/

[5] Hooks, XCLASS, Services :

http://typo3.org/documentation/document-library/core-documentation/doc_core_api/4.1.0/view/3/8/

[6] TYPO3 Forum:

<http://www.typo3.net/forum/>

[7] Core Api Documentation:

http://typo3.org/documentation/document-library/core-documentation/doc_core_api/4.0.0/view/toc/0/

[8] Funktionsreferenz:

<http://typo3.org/fileadmin/typo3api-4.0.0/classes.html>