

TYPO3

Das E-Book aus der Praxis



Thomas Esders

Dienstag, 16. Februar 2010
Version 1.3

Inhaltsverzeichnis

EINLEITUNG	3
ÜBER DIESES E-BOOK	3
TYPO3 ALLGEMEIN	4
Was brauche ich zum Einstieg?	4
Wo finde ich Hilfe, wenn ich weitere Fragen habe?	6
TYPOSCRIPT	7
Beispiel Eins: Ein Auto erzeugen	7
Beispiel Zwei: Hallo Welt	9
Beispiel Drei: Content auslesen	12
Beispiel Vier: Menüs	15
Das Schweizer Messer: stdWrap	18
Variable Menüs: optionSplit	20
Bedingungen: TypoScript Conditions	23
GIFBUILDER: Mit Masken arbeiten	25
Die TSRef richtig lesen	28
TYPO3 TIPPS	31
Was benötige ich für ein Hosting?	31
Wie geht ein Update?	33
TYPO3 Package selber bauen	34
TYPO3 Suche: Crawler einrichten	36
META-Tags auch in der Single-View	38
Uhrzeit für die Felder Start und Stop	40

TYPO3 ADMINISTRATION	41
Problem: Kein Backendzugang oder Passwort vergessen	41
Problem: Passwort für das Install-Tool vergessen	43
Problem: Extension installiert und das Backend funktioniert nicht mehr	44
Problem: Weiße Seite im Frontend – Quelltext komplett leer	45

Einleitung

Viele der Kapitel in diesem E-Book stammen in ihrer Urform aus meinem Blog <http://www.webworking-blog.de>¹. Die Inhalte stammen aus der Praxis und sind über die Jahre entstanden.

Ich hielt es für eine gute Idee, diese Artikel zu überarbeiten und gesammelt als ein E-Book heraus zu geben.

Ich werde versuchen, diese E-Book regelmäßig zu ergänzen und zu erweitern.

Vielen Dank an dieser Stelle an die Jungs von TYPO3 Blogger², die Kapitel geliefert haben.

Über dieses E-Book

Im Gegensatz zu einem klassischen Buch ist dieses E-Book als eine Sammlung von abgeschlossenen praktischen Themen rund um TYPO3 zu verstehen. Die einzelnen Kapitel dieses Buches bauen in der Regel nicht aufeinander auf. TYPO3 bildet trotzdem die lose Klammer über alle Kapitel.

Ich wünsche Euch viel Spaß beim Lesen. Wer Anmerkungen zu dem Buch hat, kann mir gerne ein Feedback³ geben.

Thomas Esders

¹ <http://www.webworking-blog.de>

² <http://typo3blogger.de>

³ <http://www.webworking-blog.de/typo3/typo3-e-book/>

TYPO3 allgemein

Was brauche ich zum Einstieg?

Immer wieder bekomme ich von Anfängern die Frage gestellt, was er wissen oder lesen muss, wenn er sich mit TYPO3 beschäftigen möchte. Da das TYPO3-Projekt sehr gut dokumentiert ist, fällt die Auswahl des richtigen Dokuments am Anfang schwer. Deswegen möchte ich euch vorab eine Übersicht der wichtigsten Informationsquellen und Dokumente geben.

Der Einstieg⁴: Erklärt gut, wie TYPO3 funktioniert, wie man es installiert, konfiguriert und administriert. Sehr gut sind in diesem Zusammenhang auch noch die Videos⁵, die einen Einstieg in die Bedienung von TYPO3 geben.

Templating und TypoScript: “Modern Template Building”⁶: Erkläre anhand eines Beispiels, wie man von einer statischen HTML-Vorlage zu einem dynamischen TYPO3 kommt. Dieses Tutorial solltet Ihr einmal durchspielen bzw. durcharbeiten.

“TypoScript by Example”⁷ erklärt anhand vieler Beispiele, wie **TypoScript** funktioniert.

⁴ http://typo3.org/documentation/document-library/tutorials/doc_tut_quickstart_de/0.0.6/view/

⁵ <http://typo3.org/documentation/videos/tutorials-v4/>

⁶ http://typo3.org/documentation/document-library/tutorials/doc_tut_temple_de/0.0.2/view/

⁷ http://typo3.org/documentation/document-library/core-documentation/doc_core_tsbyex/0.0.16/view/

Für das **Dynamisieren** benötigt man dann noch die TypoScript Sprachreferenz⁸ zum Nachschlagen.

Zum Ausprobieren von TYPO3 eignet sich am besten der TYPO3 Winstaller⁹. Dieser installiert in 5 Minuten nicht nur TYPO3 sondern auch die “Middleware” Apache, PHP, MySQL, etc. Das Paket basiert auf XAMPP.

⁸ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.0.0/view/

⁹ <http://typo3winstaller.sourceforge.net/.com/>

Wo finde ich Hilfe, wenn ich weitere Fragen habe?

In den offiziellen Mailinglisten¹⁰ gibt es immer kompetente Hilfe. Wer diese nur lesen möchte, kann sie hier in Form eines Forums¹¹ auch ansehen.

Das deutschsprachige Forum zu TYPO3¹² ist sicherlich auch noch eine Anlaufquelle.

Natürlich gibt es noch eine große Anzahl an Büchern für TYPO3¹³.

¹⁰ <http://typo3.org/community/mailling-lists/>

¹¹ <http://www.typo3-jack.net/>

¹² <http://www.typo3forum.net/forum/>

¹³ <http://www.webworking-blog.de/typo3/typo3-buch-empfehlung/>

TypoScript

In diesem Kapitel möchte ich euch eine Einführung in TypoScript, der Konfigurations-Sprache von TYPO3 geben.

Als erstes ist es wichtig zu verstehen, dass TypoScript keine Programmiersprache ist. TypoScript ist eher eine Konfigurationssprache, vergleichbar mit der Windows Registry.

TypoScript ist sehr mächtig. Hat man es erstmal verstanden, was zugegeben nicht so einfach ist, so kann ich die Ausgabe von Elementen wie z.B. einem Menü extrem clever gestalten.

Als offizielle Dokumentation führt nichts an der Sprachreferenz TSRef¹⁴ vorbei. Weiterhin gibt es noch ein Tutorial "TypoScript by Example"¹⁵, das in vielen Beispielen die Verwendung von TypoScript erklärt.

Beispiel Eins: Ein Auto erzeugen

In TypoScript bildet man so genannte Objekt-Bäume um zu definieren, wie z.B. ein Menü aussehen soll. Wir schauen und aber als erstes Beispiel ein TypoScript an, das ein Auto definiert.

```
1 lib.meinAuto = AUTO
2 lib.meinAuto.farbe = #FF0000
3 lib.meinAuto.10 = MOTOR
4 lib.meinAuto.10.typ = diesel
```

In Zeile 1 in diesem Beispiel erzeugen wir ein neues Objekt, das ich "lib.meinAuto" nenne. Der Name ist dabei frei wählbar.

Diesem Objekt weise ich die Klasse "AUTO" zu. Woher weiß ich, dass ich diese Klasse benutzen kann? In der TSRef kann man das nachlesen. Wir kommen später genauer darauf zurück.

¹⁴ http://typo3.org/documentation/document-library/references/doc_core_tsref/current/

¹⁵ http://typo3.org/documentation/document-library/core-documentation/doc_core_tsbyex/current/

Nun kann ich für das Objekt "lib.meinAuto" Eigenschaften definieren. Welche möglich sind, steht wieder in der TSRef. In unserem Fall darf ich z.B. für ein Objekt der Klasse "AUTO" die Eigenschaft "farbe" definieren (Zeile 2).

In der dritten Zeile nehme ich eine so genannte Objekterweiterung vor. Dazu nehme ich mein bestehendes Objekt "lib.meinAuto" und ergänze den Pfad um einen Zähler, in diesem Fall die 10. Ich kann dem neuen Objekt "lib.meinAuto.10" nun eine neue Klasse zuweisen. In diesem Fall die Klasse "MOTOR".

In Zeile 4 benutze ich nun die Eigenschaft "typ" der Klasse "MOTOR" um zu definieren, dass mein Auto einen Dieselmotor haben soll.

TYPO3 erzeugt nun im Frontend ein Auto, das feuerrot ist und einen Diesel als Motor hat.

Beispiel Zwei: Hallo Welt

So, genug Autos gebaut. Nun wollen wir uns mal ansehen, wie man TypoScript im wirklichen Dynamisierer-Leben benutzt. Wir wollen eine weiße Seite bauen, auf der “Hallo Welt” steht.

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.value = Hallo Welt
```

Eine Webseite benötigt einen HEAD-Bereich und einen BODY-Bereich. Diese müssen wir als erstes erzeugen. Wir schlagen also in der TSRef nach, wie man eine “leere” Seite erzeugt¹⁶.

7.5. "PAGE":

Pages are referenced by two main values. The "id" and "type".

The "id" points to the uid of the page (or the alias). Thus the page is found.

The "type" is used to define how the page should be rendered. This is primarily used with framesets. Here the frameset normally has the type=0 (or not set) and the documents in the frameset would be defined with another type, eg. type=1 for the content-page.

You should explore the framesets of the TYPO3-sites around. Also look in the standard-templates for framesets.

It's a good habit to use type=1 for the main-page of a website with frames. With no-frames sites type is normally zero.

Another good habit is to use "page" as the toplevel-objectname for the content-page on a website.

Most of this codes is executed in the PHP-script pagegen.php

Property:	Data type:	Description:	Default:
typeName	typeName	This decides the the typeId of the page. The value defaults to 0 for the first found PAGE object, but it MUST be set and be unique as soon you use more than one such object (watch this if you use frames on your page)!	0

Abbildung 1: TSRef Page Objekt

¹⁶ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/7/5/

Wir sehen eine kleine Einleitung zu dem “PAGE”-Objekt und darunter eine Tabelle. Diese Tabelle ist wie folgt zu verstehen:

Property: Dieser Spalte kann man entnehmen, was ich alles an Eigenschaften für meinen Objektpfad definieren kann, wenn ich dem Objekt die Klasse “PAGE” zugeordnet habe. In unserm Fall benutzen wir in Zeile 2 die Eigenschaft “bodyTag”.

Data Type: Hier ist definiert, was ich nach dem Gleichheitszeichen eingeben darf.

Description: Eine kurze Beschreibung.

Default: Was wird gesetzt, wenn ich nichts definiere.

Sehen wir uns unser Beispiel 2 noch einmal an.

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.value = Hallo Welt
```

In Zeile eins definiere ich ein Objekt “meineSeite” und weise ihm die Klasse “PAGE”¹⁷ zu. Damit erzeugt TYPO3 schon mal eine weiße Seite mit einem HEAD und einem BODY.

In Zeile zwei benutze ich die Eigenschaft “bodyTag” der Klasse “PAGE”, um den Body-Tag zu definieren.

In der dritten Zeile mache ich eine Objekterweiterung “meineSeite.10”, um dieser die neue Klasse “TEXT”¹⁸ zuzuweisen. Diese sorgt dafür, dass ich einen Text ausgeben kann.

¹⁷ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/7/5/

¹⁸ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/8/3/

In der vierten Zeile benutze ich die Eigenschaft “value” der Klasse “TEXT”, um den Wert eingeben zu können, der ausgegeben werden soll.

Als Ergebnis erhalten wir eine weiße Seite auf der rechts oben “Hallo Welt” steht.

Beispiel Drei: Content auslesen

Im nächsten Teil der Einführung in TYPOScript geht es darum, Inhalte die ein Redakteur erfasst hat, auf der Webseite darzustellen.

Wenn ein Redakteur auf einer Seite neue Inhalte erstellt und speichert, so werden diese in der Datenbank (Tabelle tt_content) abgelegt. Unsere Aufgabe ist es nun, diese Inhalte aus der Datenbank wieder auszulesen und mit HTML-Tags aufbereitet auf der Webseite darzustellen.

In der TYPOScript-Referenz TSRef gibt es eine Klasse, die uns das ermöglicht. Diese Klasse nennt sich "CONTENT"¹⁹.

Nun wäre es möglich, diese Klasse zu benutzen, um die Inhalte aus der Datenbank auszulesen. Es würde uns aber etwas fehlen. Wir müssen den Inhalt nämlich für die Darstellung aufbereiten und mit HTML-Tags anreichern.

Füllt der Redakteur das Überschriftsfeld aus, so sollte bei der Ausgabe auch ein <H1> – Tag um die Überschrift herum geschrieben werden.

Um uns viel Arbeit zu ersparen können wir hier auf ein so genanntes "statisches Template", also vorgefertigtes TYPOScript, zurückgreifen.

¹⁹ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/8/9/

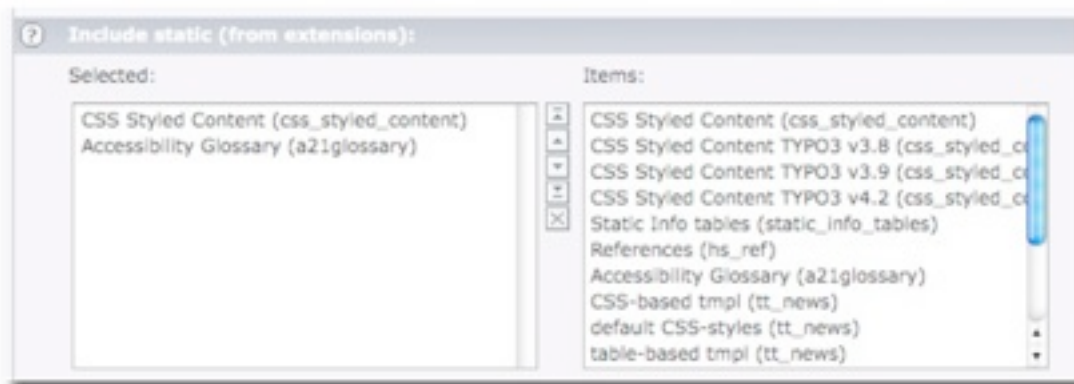


Abbildung 2: CSS Styled Content

Wenn wir in unserem Haupt-Template an die Stelle “include static (from extension)” navigieren, sehen wir rechts “CSS Styles Content (css_styles_content)”. Dieses wählen wir aus. Im Hintergrund steht uns nun viel TypoScript zur Verfügung, das wir benutzen können.

Schauen wir uns einen Teil dieses TypoScript aus CSS Styles Content genauer an:

```

1 styles.content.get = CONTENT
2 styles.content.get {
3     table = tt_content
4     select.orderBy = sorting
5     select.where = colPos=0
6     select.languageField = sys_language_uid
7 }

```

In Zeile 1 sehen wir, dass ein Objekt mit dem Namen “styles.content.get” erzeugt wird und die Klasse “CONTENT” zugewiesen wird. Dieses lädt aus der Tabelle “tt_content” Inhalte aus der Datenbank.

Wir können nun dieses Objekt bei uns benutzen. Schauen wir uns dazu das Beispiel aus dem ersten Teil noch mal an:

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.value = Hallo Welt
```

Dieses Beispiel passen wir nun entsprechend unseren Bedürfnissen an:

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 < styles.content.get
```

In Zeile drei passiert jetzt folgendes:

Unser Objekt “meineSeite.10” wird nun gefüllt durch das Objekt “styles.content.get”. Man spricht in diesem Zusammenhang auch davon, dass man TypoScript – Objekte mit Hilfe des Zeichens “<” kopieren kann.

Als Ergebnis erhalten wir nun den Inhalt der Tabelle “tt_content” für die aktuelle Seite. Das statische Template “css_styled_content” sorgt weiterhin dafür, dass der Inhalt mit HTML-Tags angereichert wird. So werden HTML-Tags um den Text geschrieben, aber auch um Bilder und deren Bildbeschreibung. Es findet das so genannte Rendering statt.

Beispiel Vier: Menüs

Erstellt ein Redakteur eine neue Seite in der Redaktionsoberfläche, so sorgt TYPO3 (genauer: TypoScript) dafür, dass ein neuer Menüpunkt an der richtigen Stelle erzeugt wird.

Wenn wir ein Menü erzeugen wollen, so benötigen wir als erstes ein Objekt dem wir die Klasse "HMENU"²⁰ zuweisen.

```
1 lib.meinMenu = HMENU
```

Als nächstes bestimmen wir, wie die Menüpunkte der ersten Ebene aussehen sollen. Dazu können wir die Eigenschaft "Zähler" (1 / 2 / 3 / ...) der Klasse "HMENU" benutzen.

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.1 =
```

Die Zahl 1 in dem Objektpfad "lib.meinMenu.1" ist dabei nicht beliebig, sondern legt fest, dass hier ein Menü der 1. Ebene definiert wird.

Doch wie soll das Menü aussehen? TYPO3 kennt unterschiedliche Menütypen. Es gibt z.B. ein reines Textmenü oder ein grafisches Menü. Der Einfachheit halber wollen wir erstmal ein einfaches Textmenü erstellen. Dazu weisen wir unserem Objektpfad "lib.meinMenu.1" die Klasse "TMENU"²¹ zu.

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.1 = TMENU
```

TypoScript kann für unterschiedliche Menüstatus²² unterschiedliche Konfigurationen vorsehen. Doch was sind Menüstatus?

²⁰ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/8/11/

²¹ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/10/7/

²² http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/10/2/

Nun ja, da gibt es als erstes den so genannte normalen Status (“NO”). Ein Menüpunkt steht da und wartet darauf angeklickt zu werden. Dann gibt es weitere Status, wie z.B. Aktiv (“ACT”). Dieser Menüpunkt ist gerade angeklickt worden. Wir wollen erstmal den Normal-Status definieren.

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.1 = TMENU
3 lib.meinMenu.1.NO = 1
```

In Zeile drei wird also definiert, dass wir den Normal-Status aktivieren wollen.

Grundsätzlich sind wir damit mit dem Menü schon durch. Im nächsten Schritt können wir das Menü nun “stylebar” machen. Die meisten Menüs bestehen aus eine unnummerierten Liste mit den HTML-Tags und . Wir müssen also dafür sorgen, dass um das ganze Menü ein geschrieben wird und dass um jeden einzelnen Menüpunkt ein geschrieben wird.

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.wrap = <ul> | </ul>
3 lib.meinMenu.1 = TMENU
4 lib.meinMenu.1.NO = 1
5 lib.meinMenu.1.NO.wrapItemAndSub = <li> | </li>
```

Was passiert hier?

In Zeile 2 benutzen wir die Eigenschaft “wrap”. Wir schauen uns das Verfahren “wrap” in einem späteren Kapitel noch genauer an.

Hier nur so viel: “Wrap” bedeutet “umfassen” und genau das haben wir hier vor. Wir wollen das Menü (das Objekt “lib.meinMenu”) von einem umfassen lassen.

In Zeile 5 sorgen wir dafür, dass die einzelnen Menüpunkte (“lib.meinMenu.1.NO”) von einem umfasst werden.

Das Ergebnis ist die Ausgabe unseres Menüs als sauber formatierte unnummerierte Liste.

Für Beispiele, wie weitere Menüs aussehen können, möchte ich hier auf das Dokument “TypoScript by Example”²³ hinweisen.

²³ http://typo3.org/documentation/document-library/core-documentation/doc_core_tsbex/0.0.16/view/7/1/

Das Schweizer Messer: stdWrap

In diesem Kapitel wollen wir uns das Schweizer Messer in TypoScript anschauen: “stdWrap”²⁴. Mein Leitspruch ist hier immer “Wenn nix mehr geht, geht stdWrap”.

Doch, wo und wozu kann ich stdWrap gebrauchen?

Schauen wir uns dazu ein Beispiel aus unserem ersten Teil noch mal an:

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.value = Hallo Welt
```

Hier haben wir eine weiße Seite erzeugt, auf der “Hallo Welt” steht. Wir haben uns dazu die Klasse “TEXT” in der TypoScript-Referenz genauer angesehen.

Eigentlich scheint dies eine recht arme Klasse zu sein. Sie besitzt nur die Eigenschaft “value”. In Wirklichkeit kann sie aber viel mehr. Wir sehen nämlich in der zweiten Zeile der Tabelle die unscheinbaren Worte “stdWrap properties”. Dies bedeutet, dass wir alle Eigenschaften von stdWrap für ein Objekt der Klasse “TEXT” definieren können.

Was kann ich damit anfangen?

Wir wollen als Beispiel versuchen, auf unsere Seite das Änderungsdatum der Seite anstatt “Hallo Welt” ausgeben zu lassen.

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.field = SYS_LASTCHANGED
```

²⁴ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/5/1/

In Zeile 4 benutzen wir die Eigenschaft “field”, um ein Feld der aktuellen Seite aus der Datenbank auszulesen. In diesem Fall nehmen wir das Feld “SYS_LASTCHANGED”. Dieses enthält das Änderungsdatum der Seite.

Das Datum wird nun im Unix-Timestamp-Format ausgegeben. Da dies für Besucher nicht gerade gut lesbar ist, wollen wir das noch etwas formatieren:

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.field = SYS_LASTCHANGED
5 meine Seite.10.date = d.m.Y
```

In Zeile 5 benutzen wir nun die Eigenschaft “date” um das Datum in dem Format tt.mm.jjjj ausgeben zu lassen.

Schließlich wollen wir noch ein “Datum:” vor dem Datum stehen haben:

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.field = SYS_LASTCHANGED
5 meineSeite.10.date = d.m.Y
6 meineSeite.10.wrap = Datum: |
```

In Zeile 6 benutzen wir die Eigenschaft “wrap” um das Objekt “meineSeite.10” von dem Wort “Datum:” umfassen zu lassen. Das “|” stellt dabei das Objekt dar. Umfassen heißt also vor dem Datum den Text “Datum:” und nach dem Datum nichts zu schreiben.

Ein paar weitere Beispiele für den Einsatz von stdWrap findet Ihr in dem Kapitel TYPO3 Tipps.

Variable Menüs: *optionSplit*

Wozu benötigt man “*optionSplit*”²⁵? Oft hat man Designs, die es vorsehen, dass in einem horizontalen Menü jeder Menüpunkt am Ende ein “-” als Trenner bekommt. Dies soll aber bei dem letzten Menüpunkt nicht der Fall sein. Ein weiteres Beispiel könnte sein, dass der erste und der letzte Menüpunkt die Schriftfarbe “rot” bekommen soll, alle mittleren Menüpunkte sollen aber “schwarz” als Schriftfarbe haben.

Schauen wir uns unser Menü aus dem vorherigen Kapitel noch einmal an:

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.wrap = <ul> | </ul>
3 lib.meinMenu.1 = TMENU
4 lib.meinMenu.1.NO = 1
5 lib.meinMenu.1.NO.wrapItemAndSub = <li> | </li>
```

Wie sollen wir hier bestimmen, dass der erste und der letzte Menüpunkt die Farbe “rot” bekommt? Der Menüpunkt wird in der Zeile 5 bestimmt, und zwar für alle Menüpunkte.

Wir benötigen also einen Mechanismus, der uns die Liste aller Menüpunkte weiter zerlegt. Hier kommt *optionSplit* ins Spiel.

Die Spielregeln von *optionSplit*

Ich kann Werte in einen ersten, einen mittleren und einen letzten Bereich einteilen:

|*|

Diese Bereiche kann ich weiter unterteilen mit:

||

²⁵ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/3/1/#id3520928

Wie wende ich die Regeln an?

Kommen wir auf unser Beispiel von oben zurück. Wir nehmen mal an, wir haben eine css-Klasse “.rot”, die eine rote Schrift macht. Wenden wir diese nun auf unser Beispiel an:

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.wrap = <ul> | </ul>
3 lib.meinMenu.1 = TMENU
4 lib.meinMenu.1.NO = 1
5 lib.meinMenu.1.NO.wrapItemAndSub = <li class=rot"> | </li>
```

Die Folge ist, dass alle Menüpunkte die rote Schrift haben. Wir wollen aber nur den ersten und den letzten Menüpunkt in rot. Also schreiben wir:

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.wrap = <ul> | </ul>
3 lib.meinMenu.1 = TMENU
4 lib.meinMenu.1.NO = 1
5 lib.meinMenu.1.NO.wrapItemAndSub = <li class="rot"> | </li> |*| <li>
| </li> |*| <li class="rot"> | </li>
```

In Zeile 5 teilen wir die einzelnen Menüpunkte in erster (E), mittlerer (M) und letzter (L) Bereich in dem wir die Bereiche mittels des Zeichens “|*|” trennen. Schematisch:

```
5 lib.meinMenu.1.NO.wrapItemAndSub = E |*| M |*| L
```


Bedingungen: TypoScript Conditions

Mittels so genannter Conditions²⁶ kann man im TypoScript auf bestimmte Ereignisse reagieren und das TypoScript anders gestalten. Als erstes einfaches Beispiel wenden wir uns einem der ersten Beispiele aus einem vorherigen Kapitel zu:

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.value = Hallo Welt
```

In diesem Beispiel wird auf einer weißen Seite “Hallo Welt” ausgegeben. Wir möchten nun “Internet Explorer” ausgeben, wenn der Besucher zum Surfen den Internet Explorer benutzt und “Firefox”, wenn er einen Firefox benutzt.

Für Conditions gibt es bestimmte “Bedingungen”, die man abfragen kann. Um den Browser des Besuchers abzufragen, benutzen wir die “Bedingung” Browser²⁷.

Das TypoScript sieht wie folgt aus:

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.value = Browser weder Internet Explorer noch Firefox
5 [browser = netscape]
6 meineSeite.10.value = Firefox
7 [browser = msie]
8 meineSeite.10.value = Internet Explorer
9 [END]
```

In Zeile 4 wird der Wert ausgegeben, wenn keine Condition wahr wird.

In Zeile 5 steht die erste Condition. Wir fragen hier ab, ob der Browser des Besuchers Netscape bzw. Firefox ist. Kann diese Frage mit ja beantwortet

²⁶ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/4/1/

²⁷ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/4/1/#id3521353

werden, so wird die Zeile 6 ausgeführt, lautet die Antwort nein, so wird zu der nächsten Condition gesprungen.

In Zeile 7 steht die nächste Condition. Ist diese wahr, so wird die Zeile 8 ausgeführt.

In Zeile 9 steht noch eine Besonderheit, die man immer am Ende einer Condition einfügen sollte. Durch [END] werden alle vorherigen Conditions "aufgehoben".

Conditions können mit einem "UND" oder "ODER" verknüpft werden, wobei ein "UND" immer Priorität hat. Ich könnte also mit einer Condition auch Kombinationen von Bedingungen abfragen:

```
3 meineSeite.10 = TEXT
4 [browser = msie] AND [version > 5]
5 meineSeite.10.value = Internet Explorer größer 5
```

Wir fragen in Zeile 4 ab, ob es sich um einen Internet Explorer in der Version größer 5 handelt.

Oft werden Conditions benutzt, um Werte übergebener URL-Parameter²⁸ abzufragen. Weiterhin ist es möglich, Umgebungsvariablen²⁹ abzufragen. Dies ist z.B. sinnvoll um "baseUrl" zu setzen, wenn der Auftritt unter verschiedenen Domänen erreichbar ist (z.B. zum testen).

Interessant ist sicherlich noch, dass man sich eine Condition auch selber programmieren³⁰ kann. Dazu kann man eine PHP Funktion erstellen, die "richtig" oder "falsch" zurück liefert um diese dann als Condition zu benutzen.

²⁸ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/4/1/#id3526924

²⁹ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/4/1/#id3526981

³⁰ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/4/1/#id3527077

GIFBUILDER: Mit Masken arbeiten

Da man es immer wieder mal brauchen kann, hier eine kleine Anleitung, wie man mit TypoScript und dem GIFBUILDER³¹ Bilder anschneiden kann. Damit lassen sich auch weitere schöne Effekte erzielen.

Das TypoScript

```
lib.mask = IMAGE
lib.mask {
    file = GIFBUILDER
    file {
        XY = 400,307
        format = png
        quality = 100
        10 = IMAGE
        10.file = fileadmin/typo3wallpaper2_03.png
        10.mask = fileadmin/mask.png
        20 = TEXT
        20.text = mit Maske I
        20.offset = 200,40
        20.fontSize = 20px
        20.fontColor = gray
    }
}
```

Und wenn man nun das folgende Bild nimmt, und die Masken variiert, kommt da folgendes bei raus.

³¹ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.2.0/view/1/9/



Abbildung 3: Original



Abbildung 4: Maske 1



Abbildung 5: Maske 2



Abbildung 6: Maske 3

Die TSRef richtig lesen

Die TSRef ist kein Buch, das man von vorne nach hinten durchließt, sondern eine Referenz zum nachschlagen. In diesem Kapitel möchte ich Euch zeigen, wie man mit der TSRef arbeitet.

Nehmen wir ein bekanntes Beispiel und arbeiten die TSRef danach ab:

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.wrap = <ul> | </ul>
3 lib.meinMenu.1 = TMENU
4 lib.meinMenu.1.NO = 1
5 lib.meinMenu.1.NO.wrapItemAndSub = <li> | </li>
```

In Zeile 1 wird ein Objekt erzeugt, dem die Klassen "HMENU" zugeordnet wird. Wir werfen also einen Blick in das Kapitel "HMENU"³².

8.11. HMENU:

Generates hierarchical menus.

Property:	Data type:	Description:	Default:
(1 / 2 / 3 /...)	menuObj	Required! Defines which menuObj that should render the menuitems on the various levels. 1 is the first level, 2 is the second level, 3 is the third level, 4 is Example: temp.sidemenu = HMENU temp.sidemenu.1 = GMENU	(no menu)

Abbildung 7: TSRef HMENU

³² http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/8/11/

In der Tabelle dieses Kapitels sehen wir nun, welche Eigenschaften wir für “HMENU” definieren können. Dies bedeutet, wir können folgendes schreiben:

```
1 lib.meinMenu = HMENU
2 lib.meinMenu.entryLevel =
3 lib.meinMenu.minItems =
```

... und so weiter. Doch was kann ich nun nach dem “=” eingeben? Dies verrät uns die zweite Spalte “Data type”. Für “entryLevel” und “minItems” steht hier jeweils “int”. Um nun herauszufinden, was “int” bedeutet, öffnen wir das Kapitel “Datatype Reference”³³. In der Tabelle suchen wir nun nach “int” und finden dort die Erklärung, dass Integer-Zahlen möglich sind.

Schauen wir uns ein weiteres Beispiel an:

```
1 meineSeite = PAGE
2 meineSeite.bodyTag = <body>
3 meineSeite.10 = TEXT
4 meineSeite.10.field = SYS_LASTCHANGED
```

In Zeile 3 weisen wir einem Objekt die Klasse “TEXT”³⁴ zu. In der Tabelle ist zu erkennen, dass neben der Eigenschaft “value” noch weitere möglich sind (“stdWrap properties”). Wir wechseln also in das Kapitel “stdWrap”³⁵. Dort ist die in unserem Beispiel verwendete Eigenschaft “field” erklärt. Sie ist vom Datentyp “fieldname” (Korrekterweise sollte der Datentyp “getText” hier stehen).

Wir schlagen also das Kapitel “Datatype Reference” auf und suchen nach “getText”. Wir sehen, dass wir für “.field” unter anderem ein Datenbankfeld “SYS_LASTCHANGED” auslesen können.

³³ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/2/2/

³⁴ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/8/3/

³⁵ http://typo3.org/documentation/document-library/references/doc_core_tsref/4.1.0/view/5/1/

Ich hoffe, ich konnte deutlich machen, wie man mit der TSRef arbeitet, um z.B. ein bestehendes TypoScript-Beispiel zu erklären. Will man selber TypoScript schreiben, funktioniert das auf die gleiche Weise. Ich suche mir eine Klasse heraus, die das kann, was ich möchte (z.B. die Klasse "TMENU") und sehe, welche Eigenschaften mir zur Verfügung stehen. Für jede Eigenschaft kann ich dann im Kapitel "Datatype Reference" nachschlagen, was ich nach dem "=" eingeben kann.

TYPO3 Tipps

Was benötige ich für ein Hosting?

TYPO3 stellt im Gegensatz zu statischen HTML-Seiten ein paar mehr Anforderungen an das Hosting. In diesem Kapitel möchte ich diese erläutern.

Webserver

Als erstes benötigt man einen Webserver. Es empfiehlt sich hier einen aktuellen Apache Webserver zu benutzen. IIS wird zwar auch unterstützt, benötigt aber etwas mehr "Pflege". Das Modul `mod_rewrite` sollte für den Einsatz von RealURL vorhanden und nutzbar sein.

PHP

TYPO3 ist in PHP geschrieben und benötigt daher PHP. Zum Einsatz sollte eine aktuelle PHP Version 5.x kommen. Ab TYPO3 Version 4.2 wird PHP 5.2 benötigt werden. Freetype sollte in PHP einkompiliert sein. Folgende Einstellungen sind für PHP in Bezug auf TYPO3 wichtig:

- `memory_limit`
- `upload_max_filesize`

PHP-Cache

Ergänzend bringt ein PHP-Cache enorme Vorteile. Es kann als Produkt eAccelerator oder Zend-Optimizer zum Einsatz kommen.

Datenbank

MySQL ist die bevorzugte Datenbank von TYPO3. Hier sollte eine aktuelle MySQL Version 5.x benutzt werden. Durch den Einsatz der Extension DBAL kann prinzipiell eine andere Datenbank verwendet werden.

Bildverarbeitung

Für die Bildverarbeitung in TYPO3, z.B. um Bilder automatisch in Dimension und Qualität kleiner zu rechnen, muss auf dem Server “GraphicsMagick” oder “ImageMagick” installiert sein. Ich empfehle den Einsatz von “GraphicsMagick”.

Weiterhin wird eine in PHP einkompilierte GDLib-Version benötigt.

Weitere Anwendungen

Je nachdem, welche Extension installiert ist, kann es sein, dass weitere Anwendungen auf dem Server installiert sein müssen. So wird für das Durchsuchen von PDFs mit der TYPO3-Suche das Tool “pdftinfo” benötigt. Details zu diesen weiteren Tools enthalten hoffentlich die entsprechenden Extension-Dokumentationen.

Hat man bei seinem Hoster diese Voraussetzungen erfüllt, so kann es auch mit TYPO3 klappen. Wer nicht einen Server sein Eigen nennt, dem kann ich nur empfehlen, einen Hoster auszusuchen, der sich mit TYPO3 auskennt. Einen Einstieg in die Auswahl kann diese TYPO3-Wiki-Seite³⁶ liefern.

³⁶ http://wiki.typo3.org/index.php/Hosting#What_works_best_for TYPO3

Wie geht ein Update?

In dem folgenden Kapitel möchte ich erklären, wie ein TYPO3 Update in der Regel durchzuführen ist. Wie bei jedem Update eines Systems empfiehlt es sich vorher zu prüfen, ob ein aktuelles Backup der Datenbank und des Dateisystems vorhanden ist.

Als erstes sollte man im Install-Tool im Abschnitt “Database Analyser” ein so genanntes “compare” machen. In diesem Zuge wird geprüft, ob das aktuelle Datenbankschema dem Soll-Zustand entspricht. Ist dies nicht der Fall, so sollten hier die benötigten Anpassungen durchgeführt werden.

Im zweiten Schritt laden wir uns die aktuellen Sourcen von typo3.org³⁷ herunter und entpacken diese auf dem Server.

Als drittes wechseln wir den Symlink “typo3_src” im Verzeichnis “htdocs” aus.

Als viertes führen wir noch einmal ein “compare” durch. Es kann nämlich sein, dass die neue TYPO3-Version Änderungen an dem Datenbankschema benötigt.

Schließlich solltet Ihr im letzten Schritt den Cache der Installation leeren und in dem Verzeichnis /typo3conf alle Dateien die mit temp_CACHED* anfangen löschen.

Danach sollten wir die neue TYPO3 Version benutzen können.

Ihr findet ein kleines Video von mir auf youtube³⁸, dass die Schritte noch einmal zeigt.

³⁷ <http://typo3.org/download/packages/>

³⁸ <http://www.youtube.com/watch?v=iTPEj3CaYOc>

TYPO3 Package selber bauen

In diesem kurzen Kapitel möchte ich erklären, wie man ein TYPO3-Package selber baut.

Wenn man TYPO3 installiert, installiert man zwei Komponenten. Zum einen die TYPO3-Sourcen und zum anderen ein so genanntes Package.

Die Sourcen enthalten dabei das eigentliche TYPO3, also z.B. die PHP-Scripte, aus denen TYPO3 besteht.

Das Package enthält den Inhalt des eigenen Auftritts. Dies ist insbesondere ein Datenbank-Dump sowie die Verzeichnisse “fileadmin”, “uploads” und “typo3conf”.

Wenn man von einem produktiven System eine Kopie erstellen möchte, weil man z.B. ein Update testen will oder eine Veränderung ausprobieren möchte, kann man sich mit ein paar einfachen Schritten ein eigenes Package des produktiven Systems erstellen:

1. In das Verzeichnis ../typo3conf wechseln.
2. mysqldump -u <db-username> -p <datenbankname> > db.sql (Damit erzeugt man einen Dump der Datenbank)
3. In das DocumentRoot wechseln
4. tar cvzf package.tar.gz fileadmin/ uploads/ typo3conf (Damit werden die Verzeichnisse "fileadmin", "uploads" und "typo3conf" zusammengetard)
5. Das Package package.tar.gz auf das Zielsystem kopieren und im DocumentRoot entpacken
6. In der Datei ../typo3conf/localconf.php die folgenden vier Einträge löschen:
 7. \$typo_db_username = 'xxx'; // Modified or inserted by TYPO3 Install Tool.
 8. typo_db_password = 'xxxxxxx'; // Modified or inserted by TYPO3 Install Tool.
 9. typo_db_host = 'xxx'; // Modified or inserted by TYPO3 Install Tool.
 10. typo_db = 'xxx';
11. Nun wie gewohnt die Adresse des Auftritts im Browser öffnen. Da wir im Punkt 6 die TYPO3-DB-Einträge entfernt haben, startet das Install-Tool im 1-2-3-Modus.

Ab hier TYPO3 wie schon bekannt installieren. Im 3. Schritt des Install-Tools aus dem DropDown den Dump aus Schritt 2. auswählen ("db.sql")

TYPO3 Suche: Crawler einrichten

Ich möchte Euch erklären, wie man die TYPO3 Extension Crawler³⁹ einsetzen kann, um die Seiten für die TYPO3-Suche automatisch indizieren zu lassen.

Normaler weise erzeugt TYPO3 beim ersten Zugriff auf eine Seite den Index für diese Seite. Hat man viele Frontend Benutzergruppen, so muss TYPO3 für jede Kombination von Benutzergruppen die ein Frontend User haben kann, einen eigenen Index der Seite anlegen.

Der Crawler ermöglicht es nun, diesen Index automatisiert zu erzeugen.

Als erstes installiert ihr die Extension Crawler.

Danach legt ihr einen Backend User “_cli_crawler” an. Dieser wird benötigt, weil wir später per Cronjob den Crawler starten wollen.

Nun konfiguriert ihr den Crawler mittels TSConfig. Dazu bearbeitet ihr die Seiteneigenschaften der obersten Seite und tragt in das Feld TSConfig folgendes ein:

```
#set up a crawl for users that arent logged in
tx_crawler.crawlerCfg.paramSets.test =
tx_crawler.crawlerCfg.paramSets.test {
    cHash = 1
    procInstrFilter = tx_indexedsearch_reindex,
tx_indexedsearch_crawler
    baseUrl = http://<meineDomain.de>/
}
#set up a crawl for users who have group id of 1
tx_crawler.crawlerCfg.paramSets.grp1 <
tx_crawler.crawlerCfg.paramSets.test
tx_crawler.crawlerCfg.paramSets.grp1{
    userGroups = 1
}
```

... usw. für jede Gruppenkombination.

³⁹ <http://typo3.org/extensions/repository/view/crawler/current/>

Damit ist der Crawler konfiguriert. Für eine Anleitung, was alles konfiguriert werden kann, schaut bitte in die Extension-Dokumentation.

Als nächstes werden wir zwei Skripte erstellen. Das erste Skript erstellt die "Crawler-Queue", definiert also, was der Crawler alles tun soll. Dieses Skript werden wir z.B. einmal die Woche aufrufen. Das zweite Skript ist dafür da, die Queue abzarbeiten. Dieses werden wir alle zwei Minuten starten.

Skript zum Queue erstellen

```
php pfad/zum/htdocs/typo3/cli_dispatch.phpsh crawler_im 1049 -d 99 -  
proc tx_indexedsearch_reindex -n 1000 -o queue
```

Die Parameter bedeuten folgendes:

- crawler_im: Sorgt dafür, dass das Tool aufgerufen wird, dass die Queue erstellt
- 1049: Die Seiten-ID, aber der gecrawled werden soll
- -d 99: Die Rekursionsstiefe
- -proc tx_indexedsearch_reindex: Es soll indiziert werden
- -n 1000: Wieviele Einträge pro Minute sollen erzeugt werden
- -o queue: Es soll die Queue erzeugt werden

Skript für das Abarbeiten der Queue

```
php pfad/zum/htdocs/typo3/cli_dispatch.phpsh crawler
```

Zum Schluss erstellen wir zwei Cron-Jobs:

- Aufrufen des ersten Skriptes z.B. einmal die Woche
- Aufrufen des zweiten Skriptes alle zwei Minuten

META-Tags auch in der Single-View

Wenn man eine TYPO3 Extension hat, die eine Listenansicht und eine Single-Ansicht hat, dann kann man zwar für die Seite, auf der das Plugin liegt, Seitentitel und Meta-Daten vergeben, aber diese sind dann bei allen Unterseite gleich. Das mögen Suchmaschinen gar nicht.

Die Lösung für dieses Problem sieht wie folgt aus:

Auf der Seite, auf der das Plugin (in diesem Beispiel die Linkliste "ab_linklist") liegt, packt man folgendes TypoScript:

```
# Condition, zieht nur an, wenn
# tx_ablinklist_pil[category_uid] als
# Parameter in der URL steht (Wert egal, deswegen "> 0"
[globalVar = GP:tx_ablinklist_pil|category_uid > 0]
lib.linkTitel=COA
lib.linkTitel {
    # Wir wollen etwas auslesen und weiterverarbeiten,
    # deswegen Klasse "Records"
    10=RECORDS
    10 {
        # Wie lautet der Parameter, der benutzt werden soll?
        # => "tx_ablinklist_pil[category_uid]"
        source = {GPvar:tx_ablinklist_pil|category_uid}
        source.insertData = 1
        # Die Tabelle, aus der der Wert gelesen werden soll
        tables = tx_ablinklist_category
        conf.tx_ablinklist_category >
        conf.tx_ablinklist_category = TEXT
        # Das Feld in der Tabelle, aus das der Wert gelesen
        # werden soll
        conf.tx_ablinklist_category.field=label
    }
}
```

Wir haben den "label" der Kategorie ausgelesen, die der Besucher gerade ansieht. Diesen können wir weiterverarbeiten (s.u.)

```
# so ähnlich wie oben, aber andere Condition...
[globalVar = GP:tx_ablinklist_pi1|uid > 0]
lib.linkTitel=COA
lib.linkTitel {
    10=RECORDS
    10 {
        # id des template-records
        source = {GPvar:tx_ablinklist_pi1|uid}
        source.insertData = 1
        tables = tx_ablinklist_link
        conf.tx_ablinklist_link >
        conf.tx_ablinklist_link = TEXT
        conf.tx_ablinklist_link.field=label
    }
}
```

Dann das Ganze auf der Seite einbauen:

```
page.headerData.17 >
# z.B. benutzen wir den oben gelesenen Wert,
# um damit "meta-description" zu setzen...
page.headerData.17 < lib.linkTitel
page.headerData.17.wrap = <meta name="description" content=" | " />
# Seitentitel
page.headerData.20 >
page.headerData.20 = COA
# ... oder den Seitentitel...
page.headerData.20.30 < lib.linkTitel
page.headerData.20.30.wrap = |-
page.headerData.20.40 = TEXT
page.headerData.20.40.data = levelfield : 1, subtitle // title
page.headerData.20.40.required = 1
page.headerData.20.wrap = <title> | </title>
```

Uhrzeit für die Felder Start und Stop

In TYPO3 ist es möglich Seiten und Seiteninhalte per Zeitsteuerung anzuzeigen bzw. zu verstecken. Dazu dienen die Felder “Start” und “Stop”.

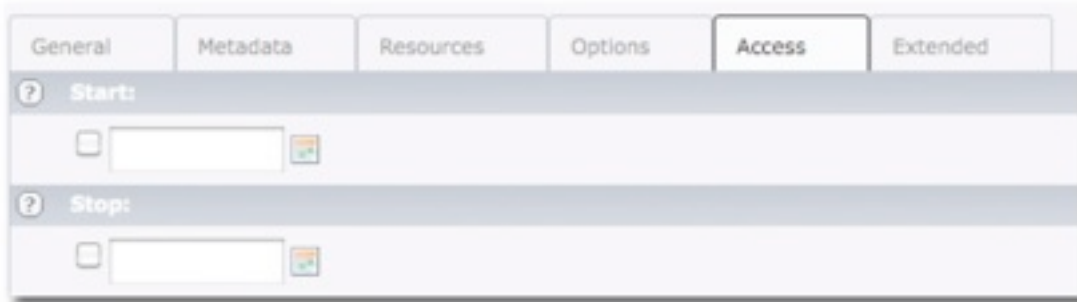


Abbildung 8: Start und Stop

In der Standardinstallation lassen sich hier aber nur Tage eingeben. Möchte man gerne auch eine Uhrzeit eingeben, so kann man dies durch eine kleine Konfiguration erreichen:

In dem Verzeichnis `.../htdocs/typo3conf/` gibt es eine Datei `“extTables.php”`. In diese einfach diese Zeilen einfügen:

```
$GLOBALS['TCA']['tt_content']['columns']['starttime']['config']['eval'] = 'datetime';
$GLOBALS['TCA']['tt_content']['columns']['endtime']['config']['eval'] = 'datetime';
$GLOBALS['TCA']['pages']['columns']['starttime']['config']['eval'] = 'datetime';
$GLOBALS['TCA']['pages']['columns']['endtime']['config']['eval'] = 'datetime';
```

Danach kann man in die Felder “Start” und “Stopp” auch Uhrzeiten eingeben. Das Format dazu ist `“11:50 20-11-2008”`.

TYPO3 Administration

Problem: Kein Backendzugang oder Passwort vergessen

Solltet Ihr das Passwort für Euren Backend Zugang verloren oder vergessen haben, so könnt Ihr Euch mit Hilfe des Install-Tools einen neuen Admin-User für das Backend anlegen.

Als erstes z.B. per FTP mit dem Server verbinden und eine leere Datei im Verzeichnis typo3conf anlegen – Name: ENABLE_INSTALL_TOOL

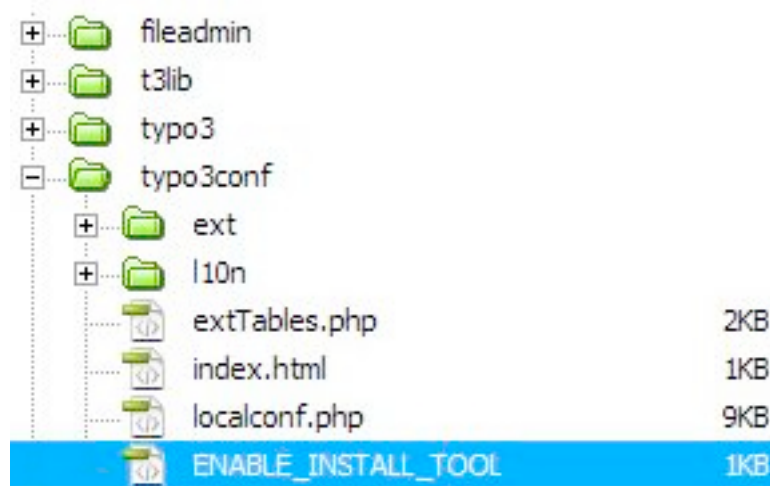


Abbildung 9: ENABLE_INSTALL_TOOL

Nun öffnet Ihr das Install-Tool: <http://www.DeineURL.org/typo3/install>

Gebt das Passwort für das Install-Tool ein (Solltet Ihr das Passwort für das Install-Tool vergessen haben -> siehe nächstes Kapitel.)

Unter dem Menüpunkt “Database Analyzer” wählt Ihr den Link “Create admin user”:

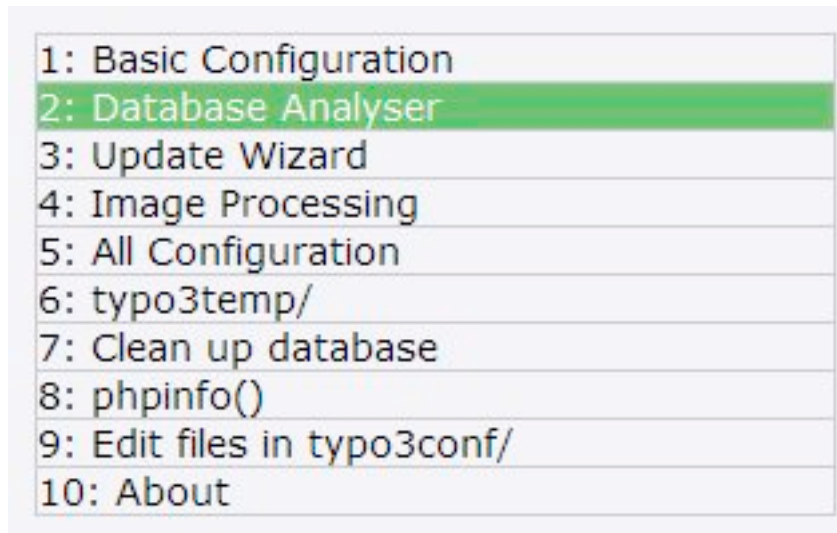


Abbildung 10: Database Analyser

Beliebigen Benutzernamen und Passwort eintragen und ein neuer Admin User ist verfügbar. Jetzt könnt ihr euch mit den neuen Zugangsdaten ins Backend einloggen.

Hinweis:

Allein an dieser Funktion seht Ihr, dass es sehr wichtig ist, das Install-Tool nach Gebrauch wieder zu deaktivieren. Löscht dazu die Datei `typo3conf/ENABLE_INSTALL_TOOL`. TYPO3 macht dies übrigens nach einer Stunde auch automatisch.

Problem: Passwort für das Install-Tool vergessen

Ihr könnt in der Datei typo3conf/localconf.php ein neues Passwort für das Install-Tool vergeben.

Via FTP oder SSH die Datei typo3conf/localconf.php öffnen und eine neue Zeile ganz unten in der Datei mit dem Wunschpasswort hinzufügen:

```
$TYPO3_CONF_VARS["BE"]["installToolPassword"] = md5('tollesPasswort');
```

Problem: Extension installiert und das Backend funktioniert nicht mehr

Wahrscheinlich macht die installierte neue Extension Probleme und muss manuell entfernt werden.

Via FTP oder SSH die Datei `typo3conf/localconf.php` öffnen und nach dem Extension Key der soeben installierten Extension suchen. Wenn ihr eine kommaseparierte Liste mit ExtKeys findet, den betroffenen Key manuell aus der Liste entfernen.

Die Liste sieht ungefähr so aus:

```
$TYPO3_CONF_VARS['EXT']['extList'] =
'css_styled_content,tsconfig_help,context_help,extra_page_cm_options,im
pexp,sys_note,tstemplate,tstemplate_ceditor,tstemplate_info,tstemplate_
objbrowser,tstemplate_analyzer,func_wizards,wizard_crpages,wizard_sortp
ages,lowlevel,install';
```

Anschließend sämtliche Cache-Dateien im Verzeichnis `typo3conf` über z.B. FTP löschen.

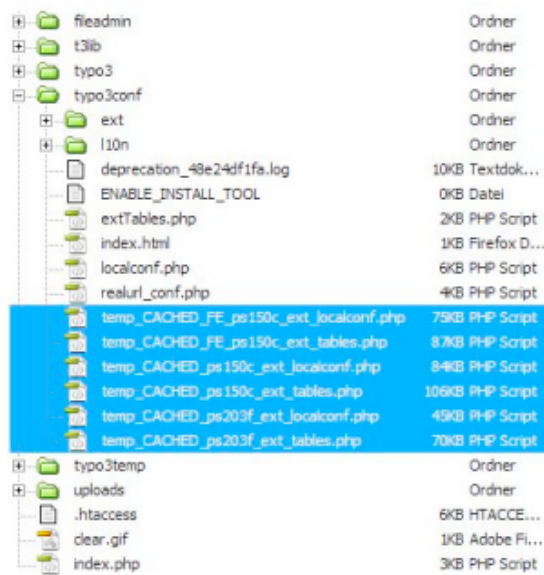


Abbildung 11: Cache-Dateien

Problem: Weiße Seite im Frontend – Quelltext komplett leer

Ruf das Install-Tool auf und aktiviere unter „All configuration“ die Option „displayErrors“.

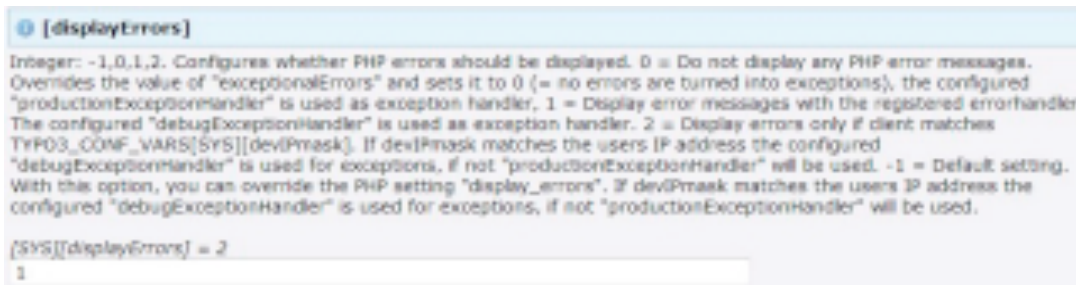


Abbildung 12: displayErrors

Trage eine 1 ein und speichere Deine Einstellungen. Jetzt sollte statt einer weißen Seite eine Fehlermeldung sichtbar sein. Die Fehlermeldung gibt Rückschlüsse auf die fehlerverursachende Extension (meist kommen die Fehler nicht aus dem Core) und natürlich auf den eigentlichen Fehler.