Grundlagenwissen Typo3 Version 3.6.2

Deutsche Typo3-Dokumentation

Dok.-Version 2.0

mittwaldmedien CM Services
Königsberger Strasse 6
32339 Espelkamp

URL: http://www.typo3server.com
eMail: info@typo3server.com

Inhaltsverzeichnis

Vorwort 7

Kapitel 1: Schritte zur ersten Seite 8

1.1 Begrifflichkeiten 8
1.2 Erster Frontend-Aufruf 8
1.3 Login in das Backend 9
1.3.1 Kurzer Überblick über die Backend-Module 10
1.3.2 Hinweis zum Modul Filelist / Dateiliste 13
1.3.3 Spache ändern 13
1.4 Eine erste Seite anlegen 14

Kapitel 2: Templates 16

2.1 Ein Template anlegen 17
2.1.1 Create template for a new site 18
2.1.2 Create an extension template 19
2.2 Info / Modify 19
2.2.1 title 19
2.2.2 Sitetitle 20
2.2.3 Description 20
2.2.4 Resources 20
2.2.5 Constants 20
2.2.6 Setup 20
2.3 "whole Template"-Record 21
2.3.1 Template löschen / Deactivate / Start / Stop 21
2.3.2 Clear Constants / Setup 21
2.3.3 Include static 22
2.4 TypoScript Object Browser 22
2.5 Template Analyser 23
2.6 Constant Editor 23
2.7 Das Typo3 Caching-Konzept 23

Kapitel 3: TypoScript Grundlagen 25

3.0 TypoScript-Syntax 25
3.0.1 Losgelöst von TypoScript 25
3.0.2 Operatoren 29
3.1 PAGE-Objekt 30
3.1.1 Die typeNum-Eigenschaft 31
3.2 TEXT-Objekt 33
3.3 TypoScript-Funktionen (stdWrap) 34
3.3.1 stdWrap 34
3.3.2 data (getText) 35
3.4: COA 36
3.5 CASE 37
3.5.1 Die Eigenschaft key (.field) 38
3.5.2 Die default-Eigenschaft 38
3.6 FILE 39

3.7 TEMPLATE 40
3.7.1 marks: Platzhalter verwenden 41
3.7.2 Designvorlagen 42
3.7.3 workOnSubpart: Teilbereiche 43
3.8 CONTENT 45
3.8.1 Vorbereitung: Seiteninhalt anlegen 45
3.8.2 Objekt CONTENT verwenden 47
3.8.2.1 tt_content 48
3.8.2.2 select : sortieren 52
3.8.2.3 select : Spalten 53
3.9 IMAGE 54
3.10 GIFBUILDER 55
3.10.1 Mit Ebenen arbeiten 56
3.10.2 offset : Positionieren 57
3.10.3 Grafischer Text 58
3.11.4 Ein einfacher Schatten 59
3.10.5 Mehr Dynamik 60
3.11 HMENU 61
3.11.1 Einführung 61
3.11.2 Vier unterschiedliche Menüarten 61
3.11.3 Fünf+ Zustände von Menüelementen 62
3.11.4 Vorbereitung: Seiten anlegen 63
3.11.5 special – was für ein Menü? 64
3.11.6 special : directory 64
3.12 TMENU 66
3.13 GMENU 67
3.13.1 Zustände einsetzen 68
3.13.2 Option Split: Elemente differenzieren 69

Kapitel 4: TypoScript Praxis 71

4.0 Vorwort 71
4.1 Geliefertes Design: Struktur Anlegen 72
4.1.1 Die geeignete Navigationsstruktur 72
4.1.2 Aufbau der Struktur im Frontend 73
4.1.3 Hilfsseiten nicht zugänglich machen 75
4.1.5 Wo befindet sich unsere Homepage? 77
4.2 Eine Designvorlage erstellen 78
4.2.1 Präzise HTML-Ausarbeitung 78
4.2.2 Grafiken & Designvorlagen 79
4.2.3 Substituieren von dynamischen Elementen 79
4.3 Umsetzung mit TypoScript 83
4.3.1 Das Projekttemplate erstellen 83
4.3.2 Seiteneigenschaften festlegen 85
4.3.3 Dateien mittels Dateimanager zur Verfügung stellen 87
4.3.4 Die Designvorlage einbinden 88
4.3.5 Die Platzhalter ansprechen: Fehleranalyse 91
4.3.7 Den Trailer erzeugen 93
4.3.7.1 Text auf den Trailer rendern 95
4.3.7.2 Den Text dynamisch darstellen 96
4.3.7.3 Eine weitere Text-Ebene hinzufügen 98

4.3.8 TMENU: Menü oben erstellen 100
4.3.9 GMENU: Das linke Menü erstellen 107
4.3.10 Eine zweite Menüebene und weitere Zustände hinzufügen 115
4.3.11 CONTENT: Inhalte ausgeben [mit content(default)] 124
4.3.11.1 Vorbereitung: Statisches Template inkludieren 124
4.3.11.2 Analyse: content (default) unter die Lupe genommen 125
4.3.11.3 Vorbereitung: Einen Seiteninhalt anlegen 129
4.3.11.4 Objekt CONTENT verwenden 130
4.3.11.5 Fehleranalyse: Es werden keine Inhalte dargestellt 130
4.3.11.6 Darstellung anpassen, Überschrift 131
4.3.11.7 Darstellung anpassen: Bodytext 134
4.3.12 Rechte Spalte: Inhalte darstellen 136
4.3.12.1 Die Spalten: colPos 136

Kapitel 5: Module und eigene Erweiterungen 137

5.0 Einführung 137
5.1 Der Typo3 Erweiterungsmanager 138
5.1.1 Shy Extensions 138
5.1.2 Die Auswahlbox "Menü" im Erweiterungsmanager 139
5.1.3 Die Spalten im Erweiterungsmanager 140
5.1.4 Detailinfos zu den Modulen 141
5.1.5 Verfügbare Module installieren 142
5.1.6 Module von der Extension Repository herunterladen 142
5.2 Generelles zur Updatefähigkeit 144
5.3 Bestehende Frontend-Module integrieren und anpassen 145
5.3.1 Das News-Modul 145
5.3.1.1 Das klassische Newsmodul installieren 145
5.3.1.2 Frontend-Plugin: Seiteninhalt / Container anlegen 146
5.3.1.3 Mögliche Codes des News-Moduls 149
5.3.1.4 Newsbeiträge erstellen 149
5.3.1.5 Frontend-Plugin: Elemente & Container 150
5.3.1.6 Das News-Modul unter die Lupe genommen 151
5.3.1.7 Kapselung Funktionalität, Konfiguration und Design 154
5.3.1.8 Das News-Modul konfigurieren 155
5.3.1.9 Die Designvorlage anpassen 158
5.3.1.10 Statische Darstellung von News 158

Kapitel 6: Diverses 159

6.1 Anpassungen mittels Conditions 159
6.2 Mehrsprachige Webseiten 160
6.2.1 Seitensprachen, Seiten und Inhalte übersetzen 160
6.2.2 TypoScript und Mehrsprachigkeit 162
6.3 Druckerfreundliche Version 164
6.3.1 Zusätzliche Parameter mitreichen (Sprache) 166
6.3.2 Besondere Darstellung von tt_content 166
6.4 Suchmaschinenfreundliche URLs 167
6.4.1 Den Webserver vorbereiten 167
6.4.2 Mit Alias-Namen arbeiten 168
6.5 Benutzerrechte Backend-Redakteure 169

6.5.1 Benutzergruppen 169
6.5.2 Benutzer anlegen (User) 171
6.5.3 Zugriffsrechte setzen 172
6.6 Statistiken mit AWStats 172

Kapitel 8: TypoScript Kurzreferenz 174

8.1 Datentypen 174
8.2 Objektgruppen 176
8.3 Funktionen / stdWrap 176
8.3.1 Daten auslesen 176
8.3.2 data (getText) 177
8.3.3 Bedingungen 178
8.3.4 Parse-Funktionen 179
8.3.5 Datums- und Zeitfunktionen 180
8.3.6 EditPanel 180
8.3.7 Debugging 180
8.3.8 imgResource 181
8.3.9 imageLinkWrap 182
8.3.10 numRows 183
8.3.11 select 183
8.3.12 split 183
8.3.13 if 184
8.3.14 encapsLines 185
8.3.15 parseFunc 186
8.4 Objekt-Referenz 187
8.4.1 PAGE 187
8.4.2 TEXT 188
8.4.3 COA 188
8.4.4 CASE 189
8.4.5 FILE 189
8.4.6 TEMPLATE 190
8.4.7 CONTENT 191
8.4.8 IMAGE 191
8.4.9 GIFBUILDER 192
8.4.10 HMENU 193
8.4.11 TMENU / TMENU_ITEM 194
8.4.12 GMENU / GMENU_ITEM 195
8.4.13 EDITPANEL 196
8.4.14 FORM 197
8.4.15 USER / USER_INT 198
8.4.16 PHP_SCRIPT / PHP_SCRIPT_INT 198
8.5 Frames 199
8.5.1 FRAME 199
8.5.2 FRAMESET 199
8.5.3 Beispiel-Framedefinition 199
8.6 OptionSplit 200
8.7 Conditions 201
8.7.1 Browser 201
8.7.2 Browser-Version 201
8.7.3 Betriebssystem 202

8.7.4 Devices 202
8.7.5 Sprache 202
8.7.6 IP-Adressen 202
8.7.7 Stunde 202
8.7.8 Minute 203
8.7.9 Wochentag 203
8.7.10 Tag des Monats 203
8.7.11 Monat 203
8.7.12 Benutzergruppe (FE) 203
8.7.13 Eingeloggter Benutzer (FE) 203
8.7.14 treeLevel 204
8.7.15 PIDInRootline 204
8.7.16 PIDupinRootline 204
8.7.17 GlobalVar / GlobalString 204
8.7.18 userFunc 204
8.8 Primäre Objekte 205
8.8.1 PAGE 205
8.8.2 CONFIG 205
8.8.3 FE_DATA / FE_TABLE 209
8.9 TSConfig : Benutzer 210
8.9.1 admPanel 210
8.9.2 options 210
8.9.3 setup 212
8.10 TSConfig : Page 213
8.10.1 mod 213

Anhang 215

Vorwort

Diese Dokumentation ist eine komplette Neuauflage der zweiten Dokumentation (Typo3 – die deutsche Dokumentation mit Referenz). Die durchweg positive Resonanz auf diese beiden Vorgängerversionen haben mich darin bestärkt, dieses Konzept erneut beizubehalten. Die Erfahrungen, die ich aufgrund über zweijähriger Erfahrung mit Typo3-Schulungen sammeln durften, sind in diese Dokumentation eingeflossen.

Die Deutsche Typo3-Dokumentation soll in erster Linie Grundlagenwissen vermitteln. Auf Modulkonfigurationen wurde beispielsweise nur bedingt eingegangen – statt dessen wird das Grundprinzip von Modulen im Allgemeinen am Beispiel des News-Moduls aufgezeigt.

Auf Anregungen, Kritik und Lob zur Deutschen Typo3-Dokumentation freue ich mich unter der eMail-Adresse dokumentationen@typo3server.com.

Ich wünsche Ihnen viel Spaß beim Erlernen von Typo3.

Ihr Robert Meyer

Kapitel 1: Schritte zur ersten Seite

1.1 Begrifflichkeiten

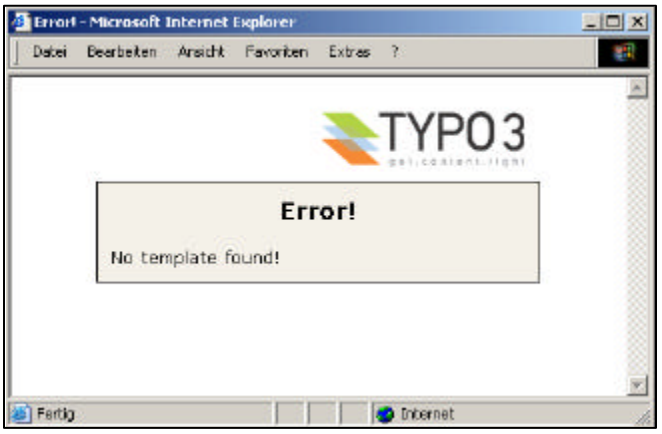
Da Typo3 überwiegend in Dänemark entwickelt wird und die gesamte Dokumentation überwiegend in der englischer Sprache verfügbar ist, versuchen wir hier, einige deutsche Begriffe einzuführen.

Bei Content Management Systemen spricht man generell von zwei Bereichen: Dem Frontend und dem Backend. Das Frontend (FE) stellt im Prinzip die reguläre Internetpräsentation dar (Website), während im Backend (BE) die Präsentation selbst erstellt und gepflegt wird. Die Begriffe Frontend und Backend mit seinen Abkürzungen FE sowie BE sollten Sie sich merken, da diese beim Verständnis von einigen Variablen und Funktionen eine wichtige Rolle spielen.

Die englischen Dokumentationen sprechen mehrdeutig von Templates. Templates können sowohl HTML-Designvorlagen als auch TypoScript-Templates sein. Dies mag zur Einführung verwirrend klingen. Wichtig ist jedoch, das Designvorlagen aus regulären HTML-Seiten bestehen, während TypoScript-Templates eben aus TypoScript, der eigenen Scriptsprache, bestehen. Um diese Mehrdeutigkeit zu umgehen, werden in diesem Buch die Begriffe Designvorlagen für HTML-Templates sowie Templates für TypoScript-Templates verwendet.

1.2 Erster Frontend-Aufruf

Vorausgesetzt Typo3 ist korrekt und einwandfrei installiert, erhalten wir beim einfachen Aufruf unserer Internetpräsentation bei einem „leeren und sauberen“ Typo3 folgende Fehlermeldung:



Diese Meldung ist weder kritisch noch eine wirkliche Fehlermeldung. Sie sagt aus, das noch keine Seiten angezeigt werden können.

1.3 Login in das Backend

Zum Anlegen neuer Seiten (unserer ersten Seite), müssen wir im Backend eingeloggt sein. Wir hängen nun an die URL im Browser, normalerweise unsere Domain, ein /typo3 an, so z.B. www.meinedomain.de/typo3, und erhalten eine Login-Seite, die in etwa wie folgt aussieht:

Hinweis zum Ordner /typo3: Der Ordner /typo3 läßt sich nicht umbenennen. Bis zur Version 3.3.0 war dieses noch durch einige Anpassungen im Quelltext möglich. Seit der Version 3.5.0 verweisen allerdings sehr viele Module auf diesen Ordner. Änderungen am Ordnernamen /typo3 würden somit sehr viele Änderungen im Quelltext erforderlich machen – Neue Module, die nicht ordnungsgemäß ("pragmatisch") entwickelt wurden, müssten zudem permanent angepasst werden. Eine Updatefähigkeit ist unter keinen Umständen mehr gewährleistet. Der Ordner /typo3 selbst stellt kein Sicherheitsrisiko dar. Möchten Sie die Sicherheit erhöhen, können Sie dieses z.B. mittels einer .htaccess-Datei erreichen.

Hinweis zum Login: Zum erfolgreichen Login in das Backend ist es erforderlich, dass Cookies auf dem jeweiligen Client aktiviert sind. Ebenfalls können private Firewalls den Login-Vorgang verhindern, da diese, je nach Konfiguration, den „Referer“ nicht mit übergeben. Sollten Sie eine Nachricht der Art erhalten, dass Sie im Installtool einen „doNoCheckReferer“-Flag setzen sollen, loggen Sie sich in das Install-Tool ein. Unter "All Configuration" können Sie diesen "Flag" setzen.

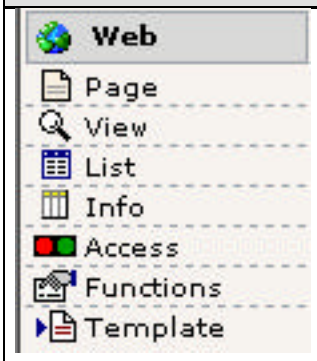
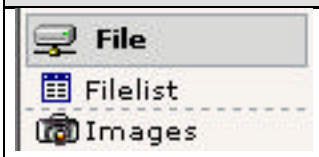
Es kann durchaus der Fall sein, dass es noch eine dritte Zeile gibt, in der man das Interface auswählen kann (Front End, Backend (Alternative), Classic).

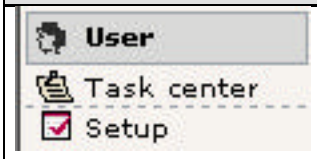
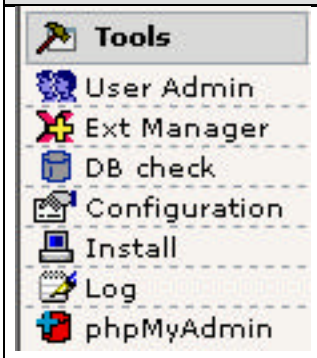
1.3.1 Kurzer Überblick über die Backend-Module

Nach erfolgtem Login als Administrator gelangen Sie in das „Alternative Backend“.

Bis einschließlich der Version 3.5.0 wurde obiges Backend als das "alternative" Backend bezeichnet. Seit der Version 3.6.0 ist die offizielle Bezeichnung nur noch "Backend".

Im Folgenden werden die einzelnen Backend-Modulbereiche vorgestellt:

Modul	Englisch	Deutsch	Beschreibung
Web			Unter dem Web-Modul wird redaktionelle gearbeitet. Ein Seitenbaum wird dargestellt.
	Page	Seite	Hier findet der strukturelle Aufbau sowie die Pflege der Internetseiten statt. Administratoren legen die Struktur fest, Redakteure können Inhalte bearbeiten.
	View	Anzeigen	Zeigt die Präsentation an. Zu vernachlässigen.
	List	Liste	Hier werden sämtliche Datensätze installierter Module angezeigt, die sich auf einer bestimmten Seite befinden. Ähnlich wie „Seite“, jedoch im Listing-Format.
	Info	Info	Zu vernachlässigen.
	Access	Zugriff	Einzelne Seiten können mit Zugriffsrechten (Besitzer, Gruppe, Alle: Lesen, Schreiben, Löschen) versehen werden. Wichtiges Modul, wenn mit Redakteuren gearbeitet werden soll.
	Functions	Funktionen	Nützliche Tools wie z.B. das Anlegen von bis zu 10 Unterseiten auf einmal.
	Template	Template	Der wohl wichtigste Bereich: Hier wird mit TypoScript gearbeitet. Zusätzlich stehen Tools wie z.B. der Objekt-Browser zur Verfügung.
File / Datei			
	Filelist	Dateiliste	Ein kleines online-„FTP-Programm“ mit beschränkten Möglichkeiten. Für einfache Uploads bzw. Änderungen jedoch gut geeignet. Bitte beachten Sie den Hinweis „Dateiliste“.
	Images	Bilder	Wie „Dateiliste“, jedoch werden hier nur Grafikdateien angezeigt. Zu vernachlässigen.

User / Benutzer			
	Task Center	Aufgaben	Unter „Aufgaben“ kann ein Benutzer sich selbst oder anderen Benutzern Aufgaben zuordnen. Insb. für die integrierte Workflow-Engine von Bedeutung. Oftmals vernachlässigbar.
	Setup	Einst.	Hier kann ein Benutzer seine Backend-Sprache einstellen sowie sein eigenes Passwort neu setzen. Weitere Einstellungen gelten insbesondere der Darstellung des Backends.
Tools			
	User Admin	Benutzer Administrator	Bei der intensiven Arbeit mit unterschiedlichen Benutzern (Redakteuren) und unterschiedlichen Rechten ist das Modul „Benutzer Administrator“ eine gute Möglichkeit, Übersicht zu behalten. Bei der Arbeit mit wenigen Redakteuren ist dieses Modul zu vernachlässigen.
	Ext Manager	Erw.-Manager	Im Erweiterungsmanager (oder Extension-Manager) können Erweiterungen wie z.B. das News-Modul hinzugefügt und installiert werden. Ebenfalls steht hier die Extension-Repository zur Verfügung. Auch können eigene Module mittels des Kickstarter-Moduls schnell eingerichtet werden.
	DB check	DB Überprüfung	Übersicht über „interne Details“ des Systems. Zu vernachlässigen.
	Configuration	Konfiguration	Hier kann Einblick in einige Typo3-internen Arrays genommen werden. Nur bei intensiver Modulentwicklung interessant, daher vernachlässigbar.
	Install	Installation	Das Typo3-Install-Tool mit der Basis-Konfiguration.
	Log	Log	Sämtliche Logins und Datensatz-änderungen werden hier festgehalten. Bei Änderungen können diese oftmals rückgängig gemacht werden.
	phpMyAdmin	phpMyAdmin	Klassisches Datenbank-Tool. Oft hilfreich, um Feldnamen einer Tabelle ausfindig zu machen.

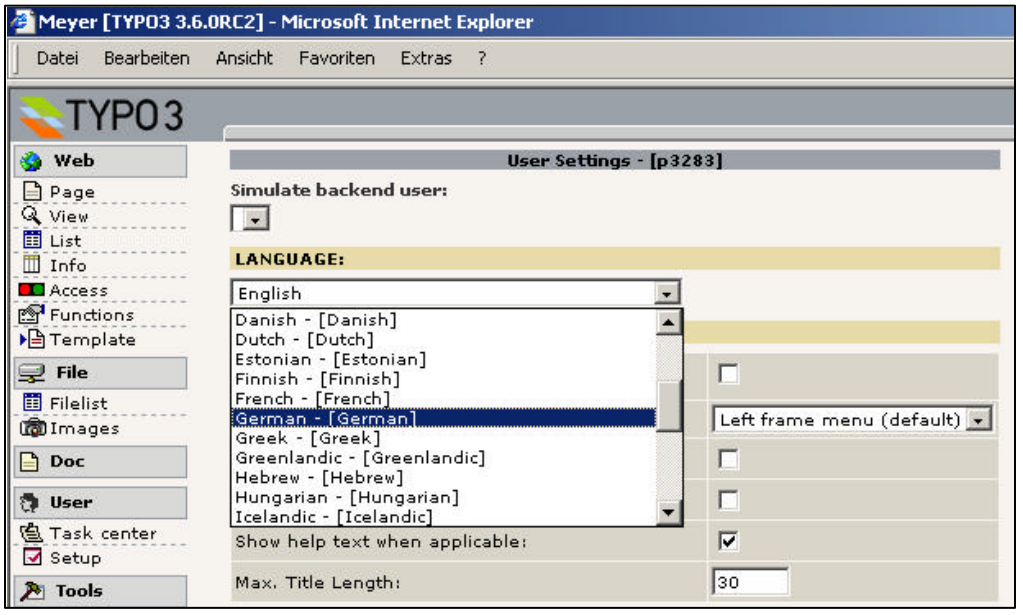
1.3.2 Hinweis zum Modul Filelist / Dateiliste

Mit diesem kleinen "FTP-Programm" können Dateien hochgeladen, modifiziert und gelöscht werden. Jedoch kann es hierbei oftmals zu Problemen mit Dateirechten kommen, da ebenfalls oftmals die Möglichkeit besteht, Dateien per FTP zu übertragen. Wird zum Beispiel eine Datei mittels FTP übertragen, gehört diese Datei in der Regel dem FTP-Benutzer. Wird eine Datei jedoch über das Backend hochgeladen, gehört diese Datei dem Benutzer „Apache“. Hierdurch können unangenehme Effekte auftreten, die die Arbeit mit Dateien und Ordnern erschweren.

Gewöhnen Sie sich deshalb möglichst an, entweder mit FTP oder mit der Dateiliste zu arbeiten. Bei auf Typo3 spezialisierten Providern wie mittwaldmedien CM Services treten diese Probleme nicht auf.

1.3.3 Spache ändern

Die erste Aktion, die wir in der Regel bei einem englischsprachigem Backend tätigen, ist die Sprache auf „Deutsch“ zu ändern. Nach erfolgreichem Login wählen wir hierzu auf der linken Seite aus dem Bereich „User“ den Menüpunkt „Settings“ aus. Auf der rechten Seite erhalten Sie im oberen Bereich eine Auswahlbox „Language“. Ändern Sie hier die Sprache von „English“ auf „German“ ab und speichern Sie Ihre Einstellungen, indem Sie ganz unten auf den Button „Save Configuration“ klicken.



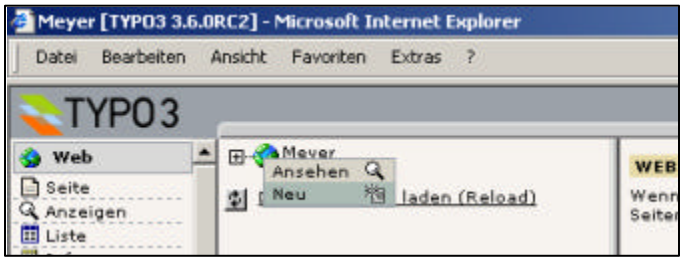
Die rechte Seite wird ab sofort in der deutschen Sprache angezeigt - das linke Menü jedoch nach wie vor in der englischen Sprache. Grund hierfür ist, dass das Backend mit Frames arbeitet und einmal aktualisiert werden muss.

Normalerweise reicht es hier aus, wenn wir nur den linken Frame aktualisieren (z.B. durch Rechtsklick mit der Maus und „Aktualisieren“ auswählen). Jedoch sollten Sie sich gleich angewöhnen, bei solchen Veränderungen das gesamte Backend bzw. das gesamte Frameset neu zu laden. Klicken Sie daher in Ihrem Browser auf das Icon „Aktualisieren“ bzw. Loggen Sie sich einmal aus und wieder ein..

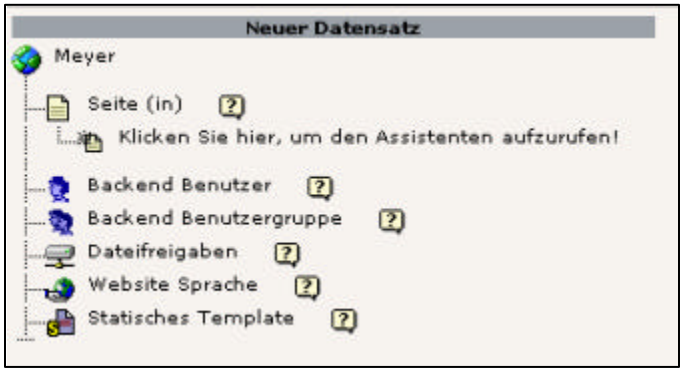
1.4 Eine erste Seite anlegen

Um eine erste Seite mit Typo3 anzulegen, klicken Sie im Menü links auf den Menüpunkt „Seite“. Es öffnet sich in der Mitte der Seitenbaum, der derzeit nur aus dem Rootlevel (Icon Weltkugel) besteht. Auf der rechten Seite öffnet sich nur ein kleiner Hinweis.

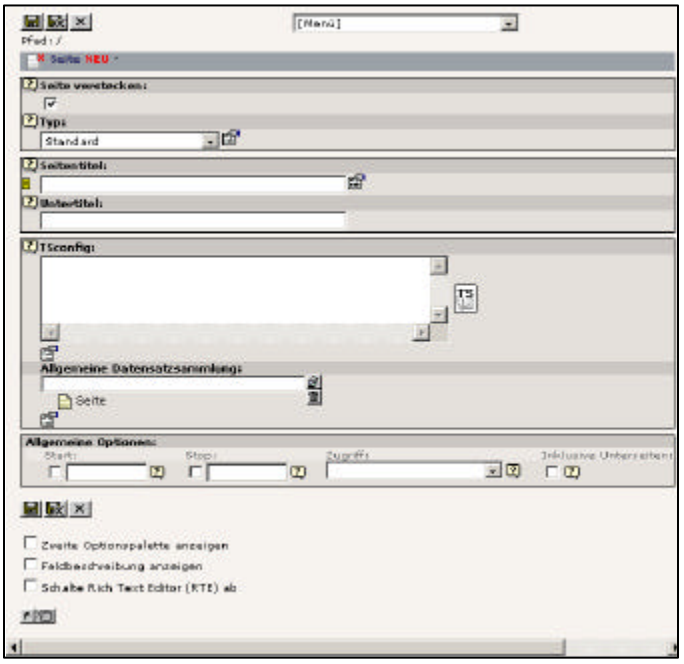
Im Seitenbaum gibt es drei Klick-Möglichkeiten: Mit dem + und dem – lässt sich der Seitenbaum öffnen bzw. schließen, um Unterseiten anzuzeigen. Ein Klick auf das Icon öffnet ein kleines Fenster mit weiteren Aktionsmöglichkeiten. Zum Anzeigen des Inhaltes kann auf den Textlink geklickt werden. Um eine neue (erste) Seite anzulegen, klicken wir auf das Icon der Rootebene (Weltkugel) und wählen aus dem Popup-Menü „Neu“ aus.



Auf der rechten Seite sehen Sie nun diverse Möglichkeiten, „etwas neues" anzulegen. Um eine neue Seite anzulegen, klicken wir auf den Textlink „Seite“ (oberster Eintrag).



Es öffnet sich rechts eine Maske, in der diverse Felder ausgefüllt werden können. Pflichtfelder werden mit einem Ausrufungszeichen links neben dem Feld gekennzeichnet. Ein solches Pflichtfeld ist hier z.B. der Seitentitel. Der Seitentitel wird unter Anderem als Bezeichner für den Seitenbaum benötigt, aber auch für den HTML-Title-Tag im Frontend, wenn die Seite aufgerufen wird.



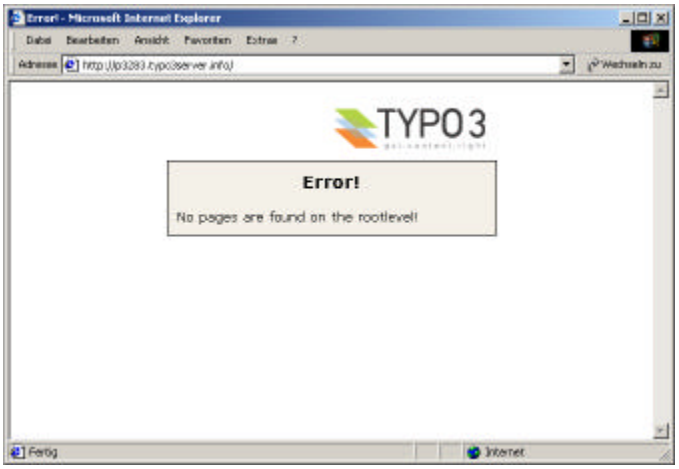
Einige Felder stehen optisch allerdings erst im Vordergrund, wenn die zweite Optionspalette aktiviert wird. Diese Einstellung können Sie aktivieren, in dem Sie bei dem Eintrag „Zweite Optionspalette anzeigen“ im unteren Teil der Seite ein Häkchen setzen.

Wir benennen unsere erste erzeugte Seite „Test“ (Angabe im Seitentitel) und speichern diese Seite ab, in dem wir das das Symbol „Speichern und Schließen“ klicken (zweites Icon oben).

Wir haben nun also unsere erste Seite ohne Seiteninhalt angelegt. Dies ist die mindeste Voraussetzung, um ein Template (TypoScript) erzeugen zu können. Templates „sagen“ Typo3, wie es was zu machen hat. Die nachfolgenden Kapitel beschäftigen sich überwiegend mit Templates und TypoScript.

Kapitel 2: Templates

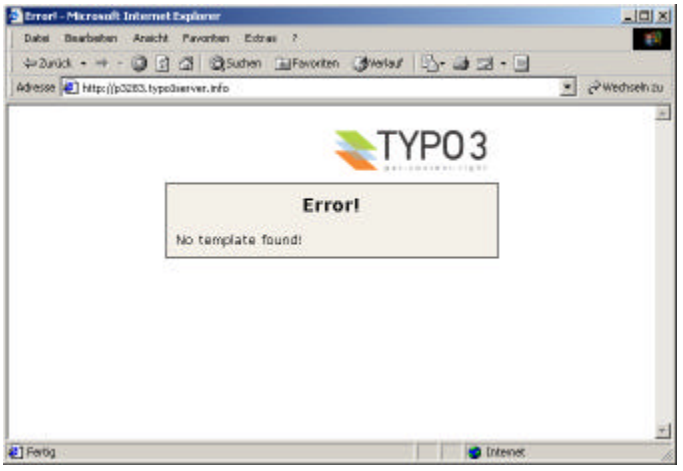
Templates teilen Typo3 mit, wie Typo3 was zu machen hat. Ohne einem Template auf einer Seite oder auf einer übergeordneten Seite weiß Typo3 nicht, dass es überhaupt etwas im Frontend darstellen soll.



Im Kapitel 1 haben wir bereits eine Seite angelegt. Ohne einer angelegten Seite erhalten wir im Frontend die Meldung „Error: No pages are found on the rootlevel“.

Templates werden immer auf Seiten angelegt und gelten für diese Seite als auch für alle untergeordneten Seiten (Template-Vererbung).

Wurde bereits eine Seite angelegt (im Kapitel 1 haben wir bereits eine Seite „Test“ angelegt), erhalten wir eine andere Fehlermeldung:



Auf der angelegten Seite kann Typo3 kein Template finden.

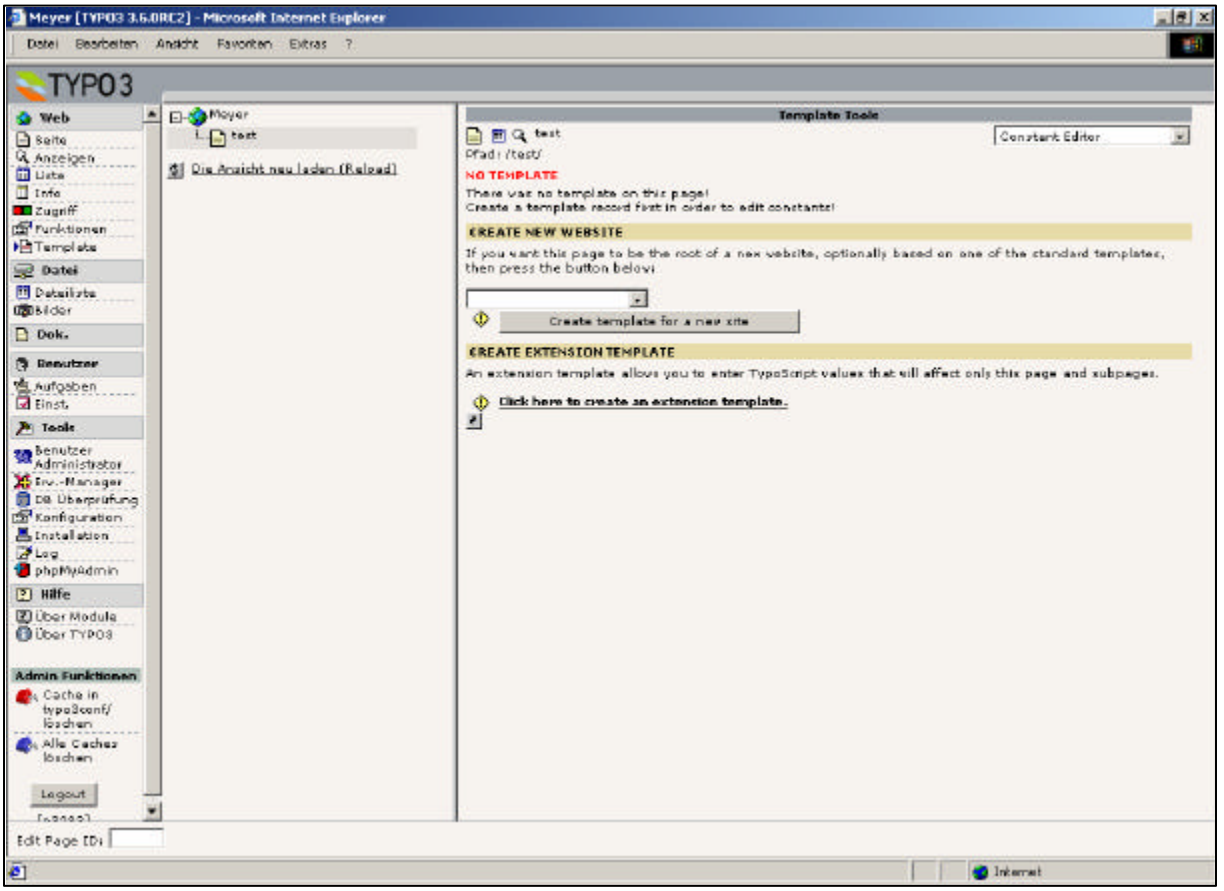
Grunsätzliches zu Templates:

- Ein Template ist ein Datensatz, der auf einer Seite liegt.
- Auf einer Seite können auch mehrere Templates liegen.
- Sämtliche Einstellungen als auch der TypoScript-Code selber werden in der Datenbank gespeichert.

2.1 Ein Template anlegen

Ein Template kann nur auf einer bestehenden Seite angelegt werden. Um ein Template anzulegen, gibt es mehrere Möglichkeiten. Eine Möglichkeit soll hier dargestellt werden:

Wir klicken im linken Menü auf "Template" und wählen in der Baumdarstellung unsere angelegte Seite (aus Kapitel 1) aus, indem wir auf den Textlink "test" klicken:

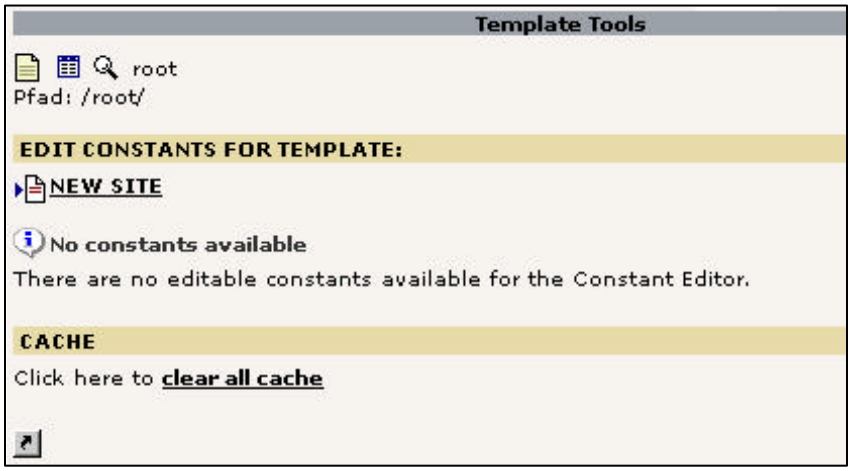


Wir erhalten den Hinweis, dass auf der aktuellen Seite "test" noch kein Template angelegt wurde ("no template").

Wir haben nun auf unserer Template-Seite folgende Möglichkeiten:

- Wir können die Auswahlbox leer lassen und auf den Button "Create template for a new site" klicken.
- Wir können aus der Auswahlbox einen Eintrag auswählen und dann auf den Button klicken.
- Wir können über den Textlink "Click here to create an extension template" ein Template anlegen.

Damit wir ein Template anlegen können, lassen wir die Auswahlbox leer und klicken auf den Button "Create template for a new site". Im rechten Frame erscheint folgende Seite:



2.1.1 Create template for a new site

Ein Klick auf den Button "Create template for a new site" legt ein Template an. Templates werden grundsätzlich immer "nach unten hin" vererbt: Das auf der Seite "test" angelegte Template gilt somit auch für Unterseiten der Seite "test".

Wenn wir über den Button ein neues Template anlegen, wird eine Vererbung "von oben" unterbrochen und es kann auf dieser Seite ein neues Projekt begonnen werden.

Aus der Auswahlbox kann ein vorgefertigtes Template ausgewählt werden. Hierunter verbergen sich unterschiedliche Designs und TypoScript-Definitionen, die angepasst werden können. Die Anpassung erfolgt mittels dem "Constant Editor" oder direkt über TypoScript.

In der Regel sollen Internet-Projekte ein individuelles Layout erhalten. Es ist daher nicht ratsam, auf diese vorgefertigten Designs zurückzugreifen. Auf eine nähere Erläuterung zur Arbeitsweise mit diesen vorgefertigten Designs wird daher verzichtet.

Wählen Sie aus der Auswahlbox keinen Eintrag aus, so wird ein neues, leeres Template erstellt. Ein auf diese Art erstelltes Template wird auch als "Projekttemplate" bezeichnet.

2.1.2 Create an extension template

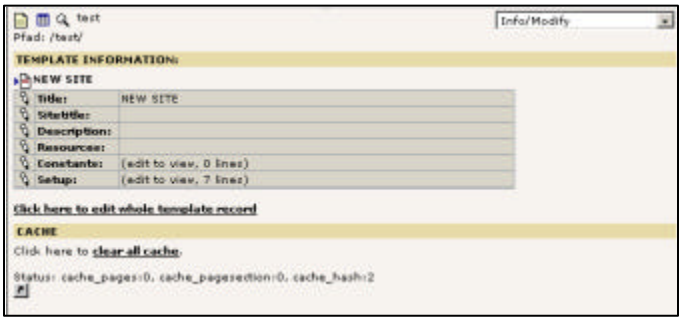
Extension-Templates bieten die Möglichkeit, vererbte Eigenschaften, Objekte, Variablen etc. zu überschreiben. Ein Template wird immer "nach unten hin" vererbt – mit einem Extension-Template wird diese Vererbung nicht unterbrochen, es können jedoch einzelne Ausnahmen definiert werden, die ebenfalls wieder "nach unten hin" vererbt werden.

Extension-Templates sollten dann angewendet werden, wenn ein Projekttemplate existiert (siehe 2.1.1), es also eine generelle Definition der Internetseite gibt, die aber ab einer bestimmten Seite anders sein soll.

2.2 Info / Modify

Rechts oben sehen wir eine Auswahlbox, die mehrere Elemente besitzt. Eines dieser Elemente ist auch "Info / Modify", mit dem wir überwiegend arbeiten werden. "Info / Modify" bietet unter Anderem die Möglichkeit, direkt mittels TypoScript die Präsentation zu beschreiben.

Wenn "Info / Modify" ausgewählt wird, steht ein "Kasten" mit mehreren Feldern zur Verfügung:



2.2.1 title

Im Feld "title" kann ein Titel für das aktuelle Template angegeben werden. Dieser Titel ist lediglich für interne Zwecke bestimmt und hat keinen Einfluss auf das Frontend.

Ein Template-Titel anzugeben findet insbesondere dann Anwendung, wenn auf einer Seite mehrere Templates vorhanden sind. Das Anlegen von mehreren Templates auf einer Seite ermöglicht, TypoScript-Code nach Themengruppen aufzuteilen - statt mit einem großen Template wird mit mehreren kleinen Templates gearbeitet. Zwecks Übersichtlichkeit sollte bei mehreren Templates auf einer Seite ein Template-Titel angegeben werden.

2.2.2 Sitetitle

Im Feld "Sitetitle" kann der Prefix für einen Seitentitel angegeben werden.

Wenn eine Seite im Frotend betrachtet wird, wird in der Regel nur der Titel der aktuellen Seite als HTML-Title-Tag aufgenommen. Soll z.B. der Firmenname immer mit erscheinen, so kann dieser Firmenname im Feld "Sitetitle" angegeben werden. Der erzeugte HTML-title-Tag wäre dann "Sitetitle:title", also z.B. "Mustermann AG : Homepage".

2.2.3 Description

Unter "Description" können Sie eine Beschreibung des Templates ablegen. Dieses Feld ist lediglich für Ihre Übersichtlichkeit bestimmt und hat keine Auswirkungen auf das Frontend.

2.2.4 Resources

Unter "Resources" können Sie Dateien in das Template einbinden. Möchten Sie z.B. mit einer bestimmten Schriftart "verdana.ttf" arbeiten, dann können Sie, sofern Sie diese Datei unter "Resources" zur Verfügung stellen, direkt mit TypoScript auf diese Datei zugreifen, ohne einen Pfad angeben zu müssen, wo sich die Datei befindet.

2.2.5 Constants

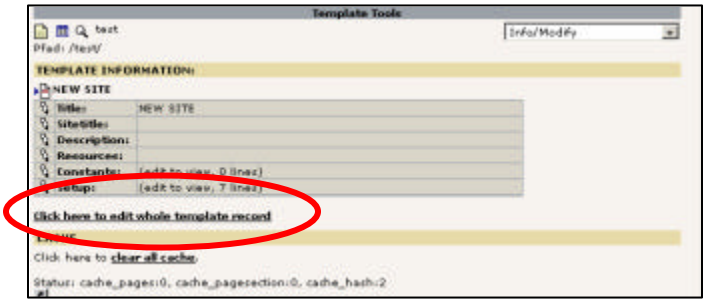
Unter "Constants" können Variablen definiert werden, die in TypoScript als Constanten ausgelesen werden können. "Constants" enthält kein TypoScript!

2.2.6 Setup

Im Feld "Setup" wird TypoScript-Code verwendet.

2.3 "whole Template"-Record

Wir finden z.B. unter "Info / Modify" einen Textlink "Click here to edit whole template record". Hier kann in das gesamte Template eingegriffen werden. Es stehen die Felder aus dem Kapitel 2.2 zur Verfügung als auch noch einige weitere, wie z.B. "Include Static".



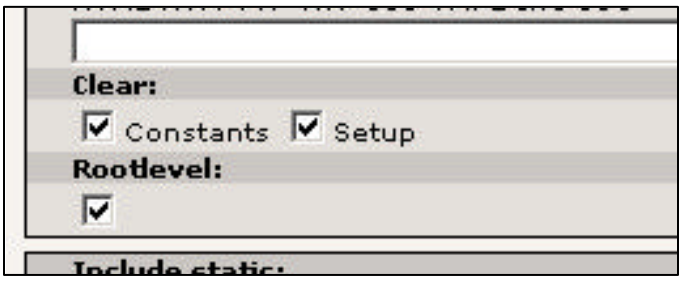
2.3.1 Template löschen / Deactivate / Start / Stop

Im oberen Abschnitt können wir beispielsweise das Template wieder löschen, in dem wir auf den "Mülleimer" klicken. Das Template wird dann in der Datenbank mit einem deleted-Flag versehen und steht in Typo3 nicht mir zur Verfügung.

Soll ein Template nur vorübergehend deaktiviert werden, ist hierfür das Feld "Deactivated" sinnvoll. Das Template wird dann im Frontend nicht mehr berücksichtigt, im Backend hingegen steht das Template noch zur Verfügung und kann zu jeder Zeit wieder aktiviert werden.

2.3.2 Clear Constants / Setup

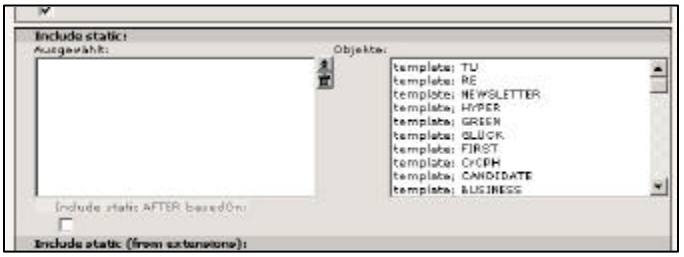
Mit den Flags "Clear Constants" / "Clear Setup" kann die Vererbung unterbrochen bzw. erlaubt werden. Ist "Clear Setup" beispielsweise aktiviert, wird von einem übergeordnetem Template (einer anderen übergeordneten Seite) das Feld "Setup" (TypoScript) nicht mehr vererbt.



2.3.3 Include static

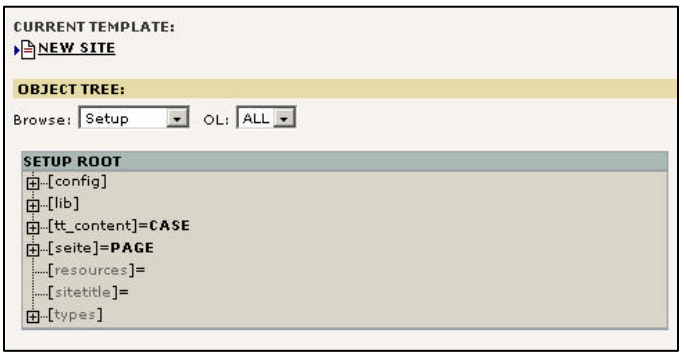
Im Feld "Include static" können sogenannte "statische Templates" inkludiert werden. In der rechten Box sehen Sie alle verfügbaren statischen Templates, in der linken Box alle eingebundenen Templates. Falls auf der rechten Seite keine Templates zur Verfügung stehen, können Sie im Install-Tool unter "Database-Analyser" -> "Import" diese statischen Templates zur Verfügung stellen.

Wir werden an späterer Stelle (Kapitel 4) das statische Template "content (default)" inkludieren. Dieses statische Template enthält eine funktionsfähige und ausgearbeitete Definition, wie Inhalte dargestellt werden können.



2.4 TypoScript Object Browser

Mit dem TypoScript-Object-Browser können wir uns durch unser TypoScript-Template "hangeln". Er ist immer dann hilfreich, wenn wir in unserem TypoScript-Code (Feld: Setup) einen Fehler haben, der nur schlecht zu identifizieren ist. Der Objekt-Browser zeigt uns den "intern aufbereitete" TypoScript-Code in Form eines Baumes an.



2.5 Template Analyser

Mit dem Template-Analyser können komplexere Template-Strukturen "analysiert" werden. So liefert der Template-Analyser zum Beispiel Einblick, in welcher Reihenfolge Template inkludiert werden. Auch kann über den Template-Analyser in den Inhalt eines inkludierten Templates (auch bei Modulen kann es vorkommen, dass ein Template inkludiert wird) eingesehen werden.

2.6 Constant Editor

Mit dem Constant-Editor können Anpassungen an Variablen bzw. Constanten getätigt werden. Voraussetzung für eine vernünftige Verwendung des Constant-Editors ist es, dass innerhalb der Constanten-Definition (Kapitel 2.2.5) zusätzliche Informationen angegeben wurden, wie diese Constanten zu editieren sind (welche Werte sind möglich etc.).

Mit dem Constant-Editor können also Variablen auf komfortable Art und Weise angepasst werden, die in TypoScript eine Verwendung finden. Von einer exzessiven Nutzung des Constant-Editors ist abzuraten, da Sie hier nur Anpassungen machen können.

2.7 Das Typo3 Caching-Konzept

Typo3 ist dynamisch. Alle Inhalte werden somit in Datenbanken gehalten und „auf Abruf“ entsprechend erzeugt. Um jedoch die Serverlast bzw. Performance zu erhöhen, wurde Caching eingeführt. Hierdurch werden Inhalte als auch vom System erzeugte Grafiken statisch gehalten, bis sich Veränderungen ergeben.

Die Datenbankabfragen zum Erzeugen einer Seite sind so komplex, das Sie für große Präsentationen mit viel Verkehr nicht ständig ausgeführt werden können. Es sind in der Regel auch immer die gleichen Daten, die der Benutzer bekommt, bis sich am dynamischen Inhalt etwas verändert hat. Hierzu speichert Typo3 seine „Ergebnisse“ einer Datenbank-Abfrage in einer Caching-Tabelle – ebenfalls in der Datenbank. Komplexe Abfragen über mehrere Tabellen sind somit nicht mehr notwendig, da Typo3 einmal gespeicherte Ergebnisse aus einer einzelnen Tabelle holen kann.

Insbesondere bei den erzeugten Grafiken führt dieses zu enormen Vorteilen. Müssten für jeden Abruf die benötigten Grafiken erneut erzeugt werden, so wäre die Serverlast ein mehrfaches und hätte Typo3 in die Schublade "langsam" katapultiert. Typo3 weiß in der Regel, wann sich eine Seite verändert hat, wann neue Grafiken berechnet werden müssen und wann Inhalte bzw. Grafiken aus den vorhandenen Quellen verwendet werden können.

Dennoch sind nicht alle Bereiche stets dynamisch möglich. Typo3 bietet sogar die Möglichkeit, das Sie das Ergebnis einer Suchanfrage grafisch darstellen. Hier ist wenig mit Caching möglich, da jede Suchanfrage in der Regel anders ausfällt und sich somit die vorhandenen Daten aus der Caching-Tabelle nicht verwenden lassen. Unterlassen Sie solche „Spielereien“ bitte zugunsten Ihres Servers und zugunsten der Geschwindigkeit Ihrer eigenen Präsentation.

Leider kann das Caching auch manchmal zu Problemen führen – nicht in Bezug auf die Geschwindigkeit, sondern vielmehr in Bezug auf die Darstellung innerhalb der Präsentation. Typo3 bemerkt, wie oben schon erwähnt, wenn sich Inhalte verändert haben; allerdings ist hierauf nicht immer Verlass. Aus diesem Grunde steht im Backend die Möglichkeit zur Verfügung, den Cache zu leeren; entweder den gesamten Cache der Präsentation oder aber nur den Cache der entsprechenden Seite.

Sollten Sie mit dem Caching von Grafiken Probleme haben, so können Sie den Inhalt des Ordners typo3temp löschen. Danach müssen Sie allerdings den gesamten Cache leeren, da Typo3 ansonsten den fertigen HTML-Quelltext aus der Datenbank nimmt und auf Grafiken zeigt, die es nicht mehr gibt. Nach dem Leeren des Caches werden die Grafiken wieder neu erstellt. Wie man aber dynamische Grafiken erstellt, erfahren Sie in den Kapitel 3.10 (GIFBUILDER) und 3.13 (GMENU).

Auch bei Designvorlagen kann es zu Caching-Problemen kommen: Ändern Sie den Inhalt einer Designvorlage, ohne Typo3 hierüber zu informieren, werden Sie die Änderungen im Frontend nicht erkennen können, da die Seiten aus dem Cache zur Verfügung gestellt werden. Löschen Sie dann manuell den Cache. Was aber genau Designvorlagen sind, erfahren Sie im Kapitel 3.7 sowie 4.2.

Kapitel 3: TypoScript Grundlagen

3.0 TypoScript-Syntax

3.0.1 Losgelöst von TypoScript

Um das Grundverständnis von TypoScript besser zu vermitteln, lösen wir uns jetzt einmal von Typo3 und stellen uns vor, wir müssten eine eigene Scriptsprache entwickeln. Diese Scriptsprache soll „Objektorientiert“ sein, in diesem Fall sei damit gemeint, das einzelne Objekte zur Laufzeit erstellt werden können. Nehmen wir uns hierfür ein einfaches Windows-Programm, das z.B. mit einer Scriptsprache gesteuert werden soll. Zu Beginn haben wir eine kleine Entwicklungsumgebung (Text-Editor) und einen Parser, der unseren Scriptcode verarbeitet.

Als Objekttypen soll unser kleines Programm Bilder (IMAGE) und Textblöcke (TEXT) kennen. Jede Zeile Scriptcode soll eine Wertzuweisung sein, also ein Gleichheitszeichen enthalten.

Bsp.-Code: meinBild = IMAGE
Hiermit erzeugen wir ein Objekt vom Typ „IMAGE“ mit dem Namen „meinBild“. Jetzt können wir auf die Eigenschaften von IMAGE-Objekten zugreifen. Eigenschaften sollen z.B. sein: die Angabe einer Datei (zum Angeben einer Grafik-Datei) und die Position des Bildes in unserer Windows-Umgebung (X,Y-Koordinaten).

```
meinBild = IMAGE
meinBild.datei = c:/meinTestprogramm/bilder/bild1.bmp
meinBild.links = 300
meinBild.oben = 100
```

Somit würde das Objekt meinBild vom Typ „IMAGE“ erzeugt, die Grafikdatei eingelesen und mittels der Eigenschaften links und oben entsprechend positioniert. Nun sind wir schreibfaul und möchten nicht immer meinBild schreiben. Daher erlauben wir nun das Ausklammern:

```
meinBild = IMAGE
meinBild {
   datei = c:/meinTestprogramm/bilder/bild1.bmp
   links = 300
   oben = 100
}
```

Nun hat spätestens der Parser ein Problem, da er diesen ScriptCode nicht mehr so einfach lesen kann. Er wird daher in einem Zwischenschritt aufbereitet und zwar in seine Ursprungsform. Alle geschweiften Klammern werden hier verarbeitet. Aus dem zweiten Beispielcode wird hier also der erste Beispielcode erstellt. Der zweite Teil ist für den Scriptersteller nur als Hilfestellung gedacht.

Sehen wir uns nun z.B. ein Text-Objekt an:

```
meinText1 = TEXT
meinText1.text = Hier steht der Text
meinText1.schrift.art = Arial
meinText1.schrift.groesse = 11
meinText1.schrift.fett = 1

meinText2 = TEXT
meinText2.text = Hier steht noch ein Text
meinText2.schrift.art = Arial
meinText2.schrift.groesse = 11
meinText2.schrift.fett = 1
```

In unserer „schreibfaulen“ Version sieht es dann etwa so aus:

```
meinText1 = TEXT
meinText1 {
   text = Hier steht der Text
   schrift {
      art = Arial
      groesse = 11
      fett = 1
   }
}
meinText2 = TEXT
meinText2 {
   text = Hier steht noch ein Text
   schrift {
      art = Arial
      groesse = 11
      fett = 1
   }
}
```

Vor dem parsen würde diese „schreibfaule“ Variante also in die erste Version ohne ausklammern konvertiert. Aber wir werden noch schreibfauler. Wie wir gesehen haben, sind die Texteigenschaften bzw. Schrifteigenschaften bei beiden Textobjekten gleich (art=Arial, groesse=1, fett=1). Um uns diesen Aufwand zu ersparen, ermöglichen wir das Kopieren von Elementen:

```
meinText1 = TEXT
meinText1 {
   text = Hier steht der Text
   schrift {
      art = Arial
      groesse = 11
      fett = 1
   }
}
meinText2 < meinText1
meinText2.text = Hier steht noch ein Text
```

Der Parser erwartet wieder „fertigen“ Code, obiges Beispiel muss also wieder aufbereitet werden. Der aufbereitete Code sieht jetzt so aus:

```
meinText1 = TEXT
meinText1.text = Hier steht der Text
meinText1.schrift .art = Arial
meinText1.schrift.groesse = 11
meinText1.schrift.fett = 1

# Hier stand vorher meinText2 < meinText1
# BEGIN DES KOPIERENS
meinText2 = TEXT
meinText2.text = Hier steht der Text
meinText2.schrift .art = Arial
meinText2.schrift.groesse = 11
meinText2.schrift.fett = 1
# ENDE DES KOPIERENS
meinText2.text = Hier steht noch ein Text
```

Und genau dieses Kopieren lässt fantastische Möglichkeiten zu.

TypoScript selbst ist recht dumm. Es bietet jedoch einige wenige Funktionen, die TypoScript zu einem undurchsichtigen Code machen können. TypoScript selbst kennt nur drei Operatoren: Zuweisen, kopieren und löschen. Auch kann Inhalt von anderen Templates „included“ (hinzugefügt) werden. Der simpelste TypoScript-Operator ist der Zuweisungsoperator. Wenn man so will, dann ist er der einzige Operator, den der Parser kennt. Der Kopier- bzw. Löschoperator macht den Code bzw. die Anwendung für den Entwickler sehr einfach. Ebenfalls unterstützend wirken geschweifte Klammern (für die Schreibfaulheit) und Kommentare. All diese Möglichkeiten werden in einem Zwischenschritt zu einem für den Parser leserlichen Code aufbereitet. Der Parser erwartet eben, wie schon oben erwähnt, den TypoScript-Code mit reinen Wertzuweisungen (Eine Wertzuweisung ist das Zuweisen eines Wertes an eine Eigenschaft, somit das Gleichheitszeichen).

Folgender Beispielcode gilt nur zur Verdeutlichung, er ist in dieser Art nicht direkt anwendbar.

```
include = datei.txt
seite = PAGE
seite {
   typeNum = 0
   10 = TEXT
   10 {
      value = Dies ist eine Testseite
      wrap < meinFontWrap
   }
}
meinFontWrap >
seite.20 = TEXT
seite.20.value = Noch ein Test
seite.20.wrap < meinFontWrap
```

Inhalt der Datei datei.txt:

```
meinFontWrap = <font face=”Arial”>|</font>
```

Der aufbereitete Quelltext sieht nun wie folgt aus:

```
meinFontWrap = <font face=”Arial”>|</font>
seite = PAGE
seite.typeNum = 0
seite.10 = TEXT
seite.10.value = Dies ist eine Testseite
seite.10.wrap = <font face=”Arial”>|</font>
seite.20 = TEXT
seite.20.value = Noch ein Test
seite.20.wrap =
```

Diese Daten kann der Parser verstehen und auswerten, da jede Zeile eine Zuweisung ist.

3.0.2 Operatoren

Operator	Beschreibung
=	Wertzuweisung Bsp: seite = PAGE seite.typeNum = 0 seite.10 = TEXT seite.10.value = HELLO WORLD
>	Lösch-Operator Der Lösch-Operator löschetalle Eigenschaften und Wertzuweisungen ab einem gegebenen Punkt: Bsp: seite = PAGE seite >
<	Kopier-Operator Der Kopier-Operator kopiert alle Eigenschaften und Wertzuweisungen ab einem gegebenen Punkt: Bsp: seite = PAGE seite.typeNum = 0 seite.10 = TEXT seite.10.value = HELLO WORLD seite.20 < seite.10

{ }	Die geschweiften Klammern dienen zur vereinfachten Schreibweise (kein wirklicher Operator).
	Bsp: seite = PAGE
	seite {
	typeNum = 0
	10 = TEXT
	10.value = HELLO WORLD
	}

( )	Die einfachen Klammern dienen zur Wertzuweisung über mehrere Zeilen (kein wirklicher Operator) Bsp: seite = PAGE seite.typeNum = 0 seite.wrap (<table> <tr> <td> | </td> <tr> </table>)
#	Kommentar. Kommentare dürfen nicht (!) mit anderen Operatoren kombiniert werden. Erlaubt: # Das ist ein Kommentar Nicht erlaubt: seite = PAGE # Dieser Kommentar ist nicht gestattet

3.1 PAGE-Objekt

Beispiel:

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.bodyTag = <body bgcolor="#CCCCCC">
```

Erläuterung zum Beispiel 1:

- In Zeile 1 wird dem Bezeichner „seite“ das Objekt „PAGE“ zugewiesen. Damit hat „seite“ PAGE-Eigenschaften erhalten (siehe PAGE-Referenz). Die typeNum ist eine PAGE-Eigenschaft, die gesetzt werden muss.
- In einem Template sollte es immer ein PAGE-Objekt mit einer typeNum = 0 geben. Der typeNum wurde weiter unten aufgrund seiner Bedeutung ein eigener Abschnitt (3.1.1) gewidmet.
- In Zeile 3 wird der Eigenschaft bodyTag (Case-Sensitiv: bodyTag wird mit einem großen „T“ geschrieben!) der gesamte HTML-Tag angegeben. bodyTag hat aber intern einen Default-Wert: <body bgcolor="#FFFFFF">. Wird die Eigenschaft bodyTag nicht gesetzt, so verwendet Typo3 den internen Default-Wert.

Beispiel:

```
01     seite = PAGE
02     seite {
03        typeNum = 0
04        bodyTag = <body bgcolor="#CCCCCC">
05     }
```

Erläuterung zum Beispiel 2:
Dieses Beispiel ist identisch zum Beispiel 1, jedoch wurde ausgeklammert. Durch das Ausklammern in Zeile 2 wird jeder weiteren Eigenschaft bis zur schließenden geschweiften Klammer das „seite“ vorangestellt. Hierdurch ergibt sich intern aus Zeile 3 „seite.typeNum = 0“.

3.1.1 Die typeNum-Eigenschaft

In einem Template kann mehr als einmal das PAGE-Objekt definiert werden. Übliche Anwendungen sind z.B. die Definition eines Designs für die „normale“ Webseite und ein Design für eine Druckversion (gleicher Inhalt, anderes Design). Ebenfalls wird die typeNum-Eigenschaft bei Seiten mit Frames verwendet (mehrere Seiten, unterschiedlicher Inhalt). Auch kann die typeNum für diverse Ausgabemedien (Browser, PDA etc.) verwendet werden.

Wie wird die typeNum eingesetzt?
Üblicherweise wird die Webseite im Frontend wie folgt aufgerufen:
Möglichkeit 1: index.php?id=123
Möglichkeit 2: 123.0.html
(Es gibt noch weitere Möglichkeiten)

Bei der Möglichkeit 1 wird nur angegeben, dass die Seite mit der eindeutigen ID 123 (Tabelle pages, Feld uid) aufgerufen werden soll. Wird keine „typeNum“ angegeben, wird intern die typeNum 0 verwendet. Hierzu sucht Typo3 im Template nach einem PAGE-Objekt, das als Wert für die typeNum eine 0 hat.

Bei der Möglichkeit 2 (123.0.html, simulateStaticDocuments) wird mit der 123 ebenfalls die eindeutige Seiten-ID angegeben, die 0 gibt aber schon die typeNum an, die verwendet werden soll.

Um mit der Möglichkeit 1 eine explizite typeNum anzugeben (ungleich 0), kann der Parameter &type= verwendet werden.
Beispiel: index.php?id=123&type=1

Beispiel:

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.bodyTag = <body bgcolor=“yellow“>
04
05     test = PAGE
06     test.typeNum = 1
07     test.bodyTag = <body bgcolor=”red”>
```

Um den Bereich „seite“ auszuführen, reicht z.B. ein Aufruf mittels „index.php?id=0“. Um den Bereich „test“ auszuführen, ist ein Aufruf mittels „index.php?id=0&type=1“ notwendig.

Anwendungsfall

Ein typischer Anwendungsfall zur Verwendung der typeNum ist die Druckversion (Abschnitt 6.3)

3.1.2 Die Nummern „10, 20, 30, ...“

An einigen Stellen in TypoScript wird man diese Nummern wiederfinden. Sie ermöglichen es, dass z.B. auf dem PAGE-Objekt in sortierter Reihenfolge mehrere Objekte abgelegt werden können (COA = Content Objekt Array, Nähere Informationen hierzu im Kapitel 3.4). Möglich sind auch Werte wie 1, 2, 3. Da jedoch diese Nummern die Sortierreihenfolge (oder Ausführungsreihenfolge, Position) angeben, werden in der Regel Zehnerschritte verwendet, um noch Platz zu haben, „falls einmal etwas vergessen wurde“ (ähnliche Handhabung wie früher unter der Programmiersprache Basic).

Beispiel 1 (nicht ausführbar):

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = IRGENDEINOBJEKT
04     seite.10.wert = 123
05     seite.20 = NOCHEINOBJEKT
06     seite.20.wert = 456
```

In der Zeile 3 wird an der Position 10 ein Objekt zugewiesen, in Zeile 5 ein anderes Objekt auf der Position 20. Somit liegen auf dem PAGE-Objekt zwei weitere Objekte, die in einer gegebenen Reihenfolge ausgegeben werden (erst die 10, dann die 20).

Beispiel 2 (nicht ausführbar)

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.35 = OBJEKT1
04     seite.35.wert = 123
05     seite.20 = OBJEKT2
06     seite.20.wert = 456
07     seite.55 = OBJEKT3
08     seite.55.wert = 789
09     seite.7 = OBJEKT4
10     seite.7.wert = 135
```

In diesem Beispiel werden mehrere Objekte auf das PAGE-Objekt „gelegt“. Die Ausführungsreihenfolge ist unabhängig von der Programmierreihenfolge: Zunächst wird das Objekt OBJEKT4 ausgeführt (Position 7), dann dass OBJEKT2 (Position 20), dann OBJEKT1 (Position 35), dann OBJEKT3 (Position 7).

3.2 TEXT-Objekt

Das TEXT-Objekt ist sehr gut geeignet, um die Mächtigkeit von Typo3-Funktionen zu erklären. Das TEXT-Objekt selbst ist recht dumm und liefert nur einen statischen Text zurück (z.B. an den Browser).

Beispiel:

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = TEXT
04     seite.10.value = Hallo Welt
```

Die Zeilen 1 und 2 wurden bereits im Kapitel 3.1 erläutert. In Zeile 3 wird an der Position 10 auf dem PAGE-Objekt das TEXT-Objekt zugewiesen. Damit hat die Position 10 TEXT-Eigenschaften. Eine Eigenschaft vom TEXT-Objekt lautet „value“ und liefert einen Wert zurück - in diesem Beispiel somit „Hallo Welt“. HTML-Code kann natürlich ebenfalls übergeben werden. Die Zeile 4 könnte somit auch so lauten:

```
04     seite.10.value = <font size=“2“>Hallo Welt</font><br>
```

Damit ist das TEXT-Objekt selbst ein recht einfaches Objekt und kann statische Code ausgeben. Durch TypoScript-Funktionen lässt sich dieses Objekt jedoch schnell und einfach um mächtige Möglichkeiten erweitern, wie z.B. Dynamik. Dieses wird im Kapitel 3.3 vorgestellt.

3.3 TypoScript-Funktionen (stdWrap)

Mit TypoScript-Funktionen lassen sich Objekte, wie z.B. das TEXT-Objekt um mächtige, dynamische Funktionalitäten erweitern. Im Folgenden werden einige Funktionen vorgestellt:

3.3.1 stdWrap

Einfache Dynamik kann z.B. mit der Funktion „field“ erreicht werden. Field ließt den Wert der aktuellen Tabelle und des aktuellen Datensatzes aus.

Beispiel

```
01   seite = PAGE
02   seite.typeNum = 0
03   seite.10 = TEXT
04   seite.10.field = title
```

In diesem Beispiel würde an der Stelle 4 der Inhalt der Datenbank-Tabelle „pages“, Feld „title“ ausgegeben werden – dynamisch. Hierzu baut Typo3 intern eine SQL-Query auf, die z.B. so aussieht (sehr stark vereinfacht):

SELECT * FROM pages WHERE uid = [aktuelle Seite]

und den Inhalt des Feldes “title” zurückliefern.

Die Tabelle „pages“ wird solange verwendet, bis wir Typo3 mitteilen, dass es mit einer anderen Tabelle arbeiten soll. Im Kapitel 3.8 (CONTENT) lernen Sie einen Zugriff auf die Tabelle "tt_content" kennen.

Zusätzlich kann eine Funktion "//" angegeben werden. "//" gibt an, dass ein zweites Feld verwendet werden soll, wenn in einem ersten Feld kein gültiger Wert enthalten ist (z.B. eine leere Zeichenkette bzw. eine 0).

Beispiel:

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = TEXT
04     seite.10.field = subtitle // title
```

Wurde kein Subtitle angegeben, wird der Titel der aktuelle Seite ausgegeben.

3.3.2 data (getText)

Mit der Eigenschaft „data“, prinzipiell eine Untereigenschaft von stdWrap, kann die Eigenschaft „field“ erweitert werden und beschränkt sich nicht nur auf Datenbankinhalte.

Beispiel 1

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = TEXT
04     seite.10.data = field:title
```

In der Zeile 4 wird der gleiche Effekt erreicht wie mit "seite.10.field = title". Die Funktion data kann aber auf der Datenbankseite noch wesentlich mehr:

Beispiel 2

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = TEXT
04     seite.10.data = DB:pages:1:title
```

Hier wird in der Zeile 4 zwar (möglicherweise) der gleiche Effekt erziehlt wie im Beispiel 1, jedoch wesentlich flexibler. DB:pages:1:title gibt an, dass aus der Datenbanktabelle pages der Datensatz mit der uid=1 (unique-ID) genommen, und das Feld „title“ dieses Datensatzes zurückgeliefert werden soll.

Beispiel 3

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = TEXT
04     seite.10.data = date:d.m.Y
```

Hier wird in der Zeile 4 das aktuelle Datum des Servers zurückgeliefert. Mit d.m.Y wird die Formatierung der Ausgabe bestimmt.

3.4: COA

Mit dem Objekt COA kann, ähnlich wie schon im Kapitel 3.1 (PAGE-Objekt) erläutert, weiter aufgesplittet werden. Statt der Zuweisung eines Objektes können somit mehrere Objekte zugewiesen werden.

Beispiel

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = COA
04     seite.10.10 = TEXT
05     seite.10.10.value = Hallo
06     seite.10.20 = TEXT
07     seite.10.20.value = Welt
```

In diesem Beispiel wird an der Position 10 des PAGE-Objektes (Zeile 3) das COA-Objekt zugewiesen. COA (Content Objekt Array) kann weitere Objekte aufnehmen und diese sortiert ausgeben. Die Sortierung innerhalb des COA-Objektes erfolgt anhand der angegebenen Nummer (z.B. 10, 20). Nähere Informationen im Kapitel 3.1.

In der Praxis wird dieses Objekt sehr häufig eingesetzt. Anwendungsfälle folgen im Praxis-Kapitel 4.

3.5 CASE

Mit dem CASE-Objekt kann eine “Wenn / Dann” abfrage erstellt werden und ist vergleichbar mit einer Case-Abfrage von Programmiersprachen.

Beispiel 1

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = CASE
04     seite.10.key.field = title
05     seite.10.test = TEXT
06     seite.10.test.value = Im Title steht TEST
07     seite.10.default = TEXT
08     seite.10.default.field = title
```

- Mit key.field (Zeile 4) wird angegeben, aus welchem Datenbankfeld ein Wert entnommen werden soll. Im Beispiel 1 somit aus dem Datenbankfeld „title“.
- In der Zeile 5 wird angegebene, dass, wenn in dem Feld title „test“ steht (Groß- und Kleinschreibung beachten), ein TEXT-Objekt erzeugt werden soll.
- Die Ausgabe soll dann lauten: „Im Title steht TEST“ (Zeile 6).
- Wurde jedoch keine Übereinstimmung gefunden, dann soll der Titel direkt ausgegeben werden (Eigenschaft "default", Zeilen 7+8).

Hinweis

In der Praxis werden wir selten auf Zeichenketten hin überprüfen, da diese nicht besonders gut zur Überprüfung geeignet sind. So können z.B. Konflikte mit internen Bezeichnern auftreten oder aber die Groß- und Kleinschreibung für Verwirrung sorgen. Oftmals wird dieses Objekt in Verbindung mit Auswahlboxen gebracht, die im Backend vorzufinden sind. In einer solchen Auswahlbox, z.B. Layout mit den Elementen Layout1, Layout2 etc., werden die ausgewählten Daten in der Regel als numerischer Wert in der Datenbank abgelegt. Diese numerischen Daten sind unkompliziert in der späteren Verarbeitung, z.B. mit dem CASE-Objekt.

3.5.1 Die Eigenschaft key (.field)

Mit der Eigenschaft "key" wird der Wert angegeben, nach dem hin überprüft werden soll. "key" kann direkt ein Wert zugewiesen werden, jedoch macht dieses in der Praxis wenig Sinn.

Beispiel 2

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = CASE
04     seite.10.key = hallo
05     seite.10.hallo = TEXT
06     seite.10.hallo.value = Hier steht HALLO
```

Bei "key" können aber die im Kapitel 3.3 vorgestellten Funktionen angewendet werden, so z.B. die Funktion „.field“, dass in der Praxis sehr häufig eingesetzt wird. Aber auch andere Funktionen, wie z.B. key.data = DB:pages:1:title sind natürlich anwendbar.

3.5.2 Die default-Eigenschaft

Trifft keines der angegebenen Werte zu, wird die default-Eigenschaft verwendet, sofern angegeben. Auch der default-Eigenschaft kann ein beliebiges Objekt zugewiesen werden.

Beispiel 3

```
01     seite = PAGE
02     seite.typeNum = 0
03
04     seite.10 = CASE
05     seite.10 {
06        key.field = title
07
08        test = COA
09        test.10 = TEXT
10        test.10.value = Der Titel der Seite lautet:<br>
11        test.20 = TEXT
12        test.20.value = <b>TEST</b>
13
14        default = COA
15        default.10 = TEXT
16        default.10.value = Der angegebene Titel ist unbekannt:
17        default.20 = TEXT
18        default.20.field = title
19     }
```

- In der Zeile 4 wird an der Position 10 des PAGE-Objektes eine CASE-Abfrage ausgeführt.
- Diese Abfrage soll dynamisch auf dem Datenbankfeld „title“ stattfinden (Zeile 6).
- Steht im Datenbankfeld der Inhalt „test“, so wird in Zeile 8 ein COA-Objekt gestartet, dass wiederum 2 TEXT-Objekte enthält (Zeile 9 + 11).
- Wurde keine Übereinstimmung gefunden, der Inhalt des Datenbankfeldes ist also nicht „test“, wird die default-Eigenschaft in Zeile 14 angesprochen, die ebenfalls 2 TEXT-Objekte enthält.

3.6 FILE

Mit dem Objekt FILE kann der Inhalt einer Datei direkt zurückgeliefert werden.

Beispiel 1

Wir erstellen eine Datei “test.txt” mit folgendem HTML-Inhalt und legen diese Datei im Ordner fileadmin ab:

```
<table border="1">
  <tr>
    <td>
       HALLO WELT
    </td>
  </tr>
</table>
```

Im Feld „Setup“ unseres Templates tragen wir folgenden TypoScript-Code ein:

```
01     seite = PAGE
02     seite.typeNum = 0
03
04     seite.10 = FILE
05     seite.10 {
06        file = fileadmin/test.txt
07     }
```

Als Ergebnis sollte folgendes im Frontend sichbar sein:

Das FILE-Objekt kann damit den Inhalt aus einer Datei zurückliefern, ohne diesen jedoch manipulieren zu können. Um Manipulationen zu ermöglichen, muss das Objekt TEMPLATE zwischengeschaltet werden (Kapitel 3.7).

3.7 TEMPLATE

Das Objekt "TEMPLATE" ist ideal dafür geeignet, um in einer Datei (Objekt FILE) Änderungen vornehmen zu können. Dies findet insbesondere bei Designvorlagen Anwendung, in der bestimmte Bereiche mit Platzhaltern gekennzeichnet, und dynamisch von Typo3 ersetzt werden.

Beispiel 1

In diesem Beispiel wird die Datei „test.txt“ verwendet, die in Kapitel 3.6 erstellt wurde.

```
01     seite = PAGE
02     seite.typeNum = 0
03
04     seite.10 = TEMPLATE
05     seite.10 {
06        template = FILE
07        template.file = fileadmin/test.txt
08     }
```

Das Frontend-Ergebnis ist identisch mit dem Beispiel von Kapitel 3.6. In TypoScript können wir jedoch erkennen, dass ein Objekt „TEMPLATE“ zwischengeschaltet wurde. Dieses Objekt „TEMPLATE“ hat die Möglichkeit, Platzhalter zu ersetzen.

3.7.1 marks: Platzhalter verwenden

Beispiel 2

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = TEMPLATE
04     seite.10 {
05          template = TEXT
06          template.value = Mein Name ist ###NAME###
07     }
```

Beispiel 2 gibt anstelle des Inhaltes einer Datei eine direkt in TypoScript angegebenen Zeichenkette aus. Das Ergebnis lässt jedoch noch immer keine Manipulation erkennen. Die Eigenschaft „marks“ macht dieses aber möglich.

Beispiel 3

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = TEMPLATE
04     seite.10 {
05        template = TEXT
06        template.value = Mein Name ist ###NAME###
07        marks.NAME = TEXT
08        marks.NAME.value = Robert Meyer
09     }
```

Jetzt wird der Platzhalter ###NAME### ersetzt – in unserem Fall durch das TEXT-Objekt in Zeile 7.

Platzhalter bzw. Marker werden immer mit 3 Rautenzeichen eingeleitet und mit 3 Rautezeichen beendet (z.B. ###MARKER###).
Der Bezeichner sollte immer in Großbuchstaben geschrieben werden. Dieses ist kein zwingende Typo3-Vorgabe, jedoch wird z.B. in Modulen, die eine Designvorlage mitliefern, diese Groß-Schreibweise verwendet. Ebenfalls erleichtert es die Fehlersuche ungemein, wenn Sie Ihr Template anderen (aus welchen Gründen auch immer) zur Verfügung stellen.
Platzhalter sollten nicht in Kommentaren stehen!

3.7.2 Designvorlagen

Selbstverständlich werden in der Praxis keine HTML-Codierungen mit großem Umfang direkt als TypoScript-Werte angegeben, sondern aus einer HTML-Datei geladen. HTML-Dateien, die mit Platzhaltern versehen sind, werden als Designvorlage bezeichnet.

Um eine Designvorlage zu erstellen, muss eine solche zunächst vorhanden sein.

Beispiel 4

Die bestehende Datei „test.txt“, die im Kapitel 3.6 erstellt wurde, wird wie folgt abgeändert:

```
<table border="1">
  <tr>
    <td>Seitentitel:</td>
    <td>Datum:</td>
  </tr>
  <tr>
    <td>###SEITE###</td>
    <td>###DATUM###</td>
  </tr>
</table>
```

TypoScript:

```
01     seite = PAGE
02     seite.typeNum = 0
03
04     seite.10 = TEMPLATE
05     seite.10 {
06        template = FILE
07        template.file = fileadmin/test.txt
08
09        marks.SEITE = TEXT
10        marks.SEITE.field = title
```

```
11
12        marks.DATUM = TEXT
13        marks.DATUM.data = date:d.m.Y
14    }
```

Ergebnis zum Beispiel 4

3.7.3 workOnSubpart: Teilbereiche

Das Objekt „TEMPLATE“ bietet zudem noch die Möglichkeit, nur mit einem Teilbereich eines zurückgelieferten Dokumentes zu arbeiten. Eingeschlossen wird ein solcher Teilbereich mit sogenannten Subparts. Subparts können in der Designvorlage von der Syntax her identisch mit Markern sein. Die entsprechende TypoScript-Eigenschaft lautet workOnSubpart.

```
Dieser Text steht außerhalb des Subparts
###BEREICH###
<table border="1">
  <tr>
    <td>Seitentitel:</td>
    <td>Datum:</td>
  </tr>
  <tr>
    <td>###SEITE###</td>
    <td>###DATUM###</td>
  </tr>
</table>
###BEREICH###
Dieser Text steht außerhalb des Subparts
```

Zur Kennzeichnung, dass es sich jedoch um einen Teilbereich handelt, und nicht um Marker, dürfen die Bezeichner in Kommentaren stehen. Marker dürfen hingegen nicht in Kommentaren stehen (bzw. dies würde keinen Sinn machen).

```
Dieser Text steht ausserhalb des Subparts
<!-- ###BEREICH### begin -->
<table border="1">
  <tr>
    <td>Seitentitel:</td>
    <td>Datum:</td>
  </tr>
  <tr>
    <td>###SEITE###</td>
    <td>###DATUM###</td>
  </tr>
</table>
<!-- ###BEREICH### end -->
Dieser Text steht ausserhalb des Subparts
```

In TypoScript wird ein Subpart wie folgt angesprochen (Zeile 8):

```
01     seite = PAGE
02     seite.typeNum = 0
03
04     seite.10 = TEMPLATE
05     seite.10 {
06        template = FILE
07        template.file = fileadmin/test.txt
08        workOnSubpart = BEREICH
09
10        marks.SEITE = TEXT
11        marks.SEITE.field = title
12
13        marks.DATUM = TEXT
14        marks.DATUM.data = date:d.m.Y
15     }
```

3.8 CONTENT

Mit dem Objekt CONTENT können Datensätze in Form einer Schleife ausgegeben werden. Typo3 baut hierzu intern eine SQL-Abfrage auf. Resultat sind im Regelfall alle Datensätze, die auf der aktuellen Seite liegen, die weder versteckt noch gelöscht wurden und die mit den aktuellen Zugriffsrechten übereinstimmen.

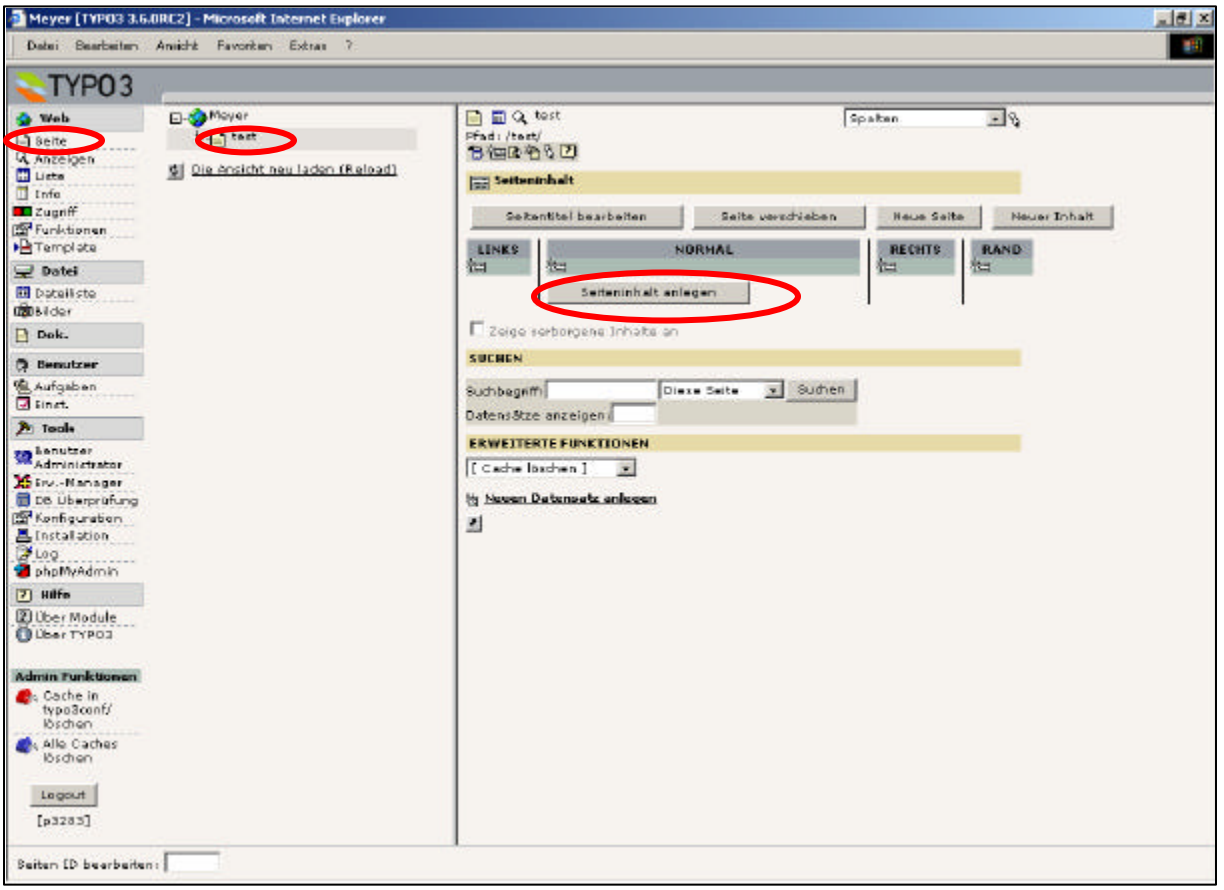
Wird z.B. in Typo3 ein Seiteninhalt angelegt, so wird dieser Datensatz in der Datenbanktabelle tt_content gespeichert. Jeder Datensatz verfügt über einen Eintrag „pid“ ("parent-ID" = Elternobjekt) und gibt an, auf welcher Seite dieser Seiteninhalt liegen soll. Das Feld „pid“ ist ein Verweis auf die entsprechenden uid (Unique-ID = Eindeutige Nummer) der Datenbanktabelle „pages“.

Beim Objekt CONTENT muss zwingend angegeben werden, von welcher Tabelle die Inhalte genommen werden sollen. Später gibt es optional die Möglichkeit, in die SQL-Abfrage einzugreifen, was durchaus Sinn macht.

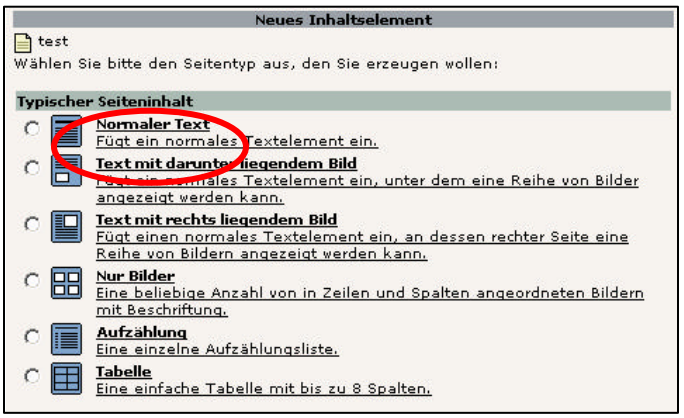
3.8.1 Vorbereitung: Seiteninhalt anlegen

Bevor wir überhaupt Inhalte ausgeben können (weitere „Stolpersteine“ werden noch folgen...), müssen wir zunächst einen Seiteninhalt anlegen. Diesen Seiteninhalt legen wir auf unserer bestehenden Seite „test“ an.

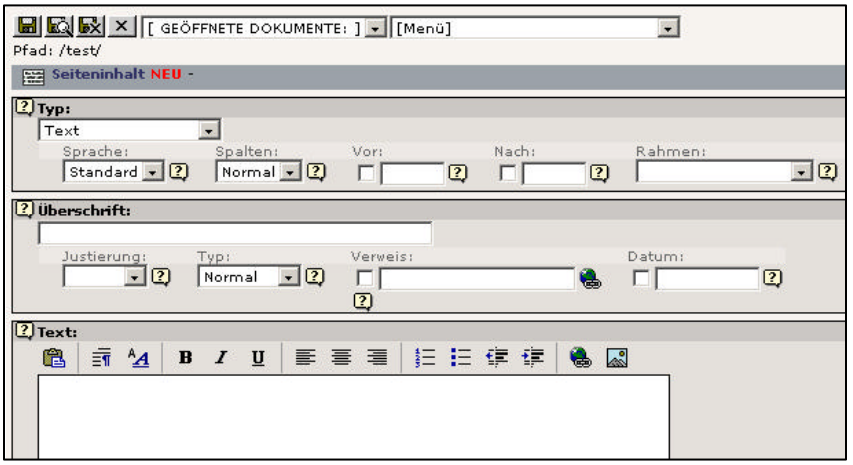
Hierzu klicken wir im linken Menüpunkt auf „Seite“ und wählen im Seitenbaum unsere Seite „test“ aus (durch anklicken des Textlinks). Auf der rechten Seite können Sie jetzt einen Seiteninhalt anlegen, in dem Sie auf den Button „Seiteninhalt anlegen“ klicken.



Mit dem Wizard für neue Inhaltselemente bzw. Seiteninhalte können Sie verschiedene vordefinierte Inhaltstypen auswählen. Für unser Beispiel wählen wir den ersten Eintrag „Normaler Text“ aus.



Füllen Sie jetzt die Masken „Überschrift“ sowie „Text“ mit der Überschrift „Dies ist die Überschrift“ und dem Text „Dies ist der Bodytext“, und speichern Sie den Seiteninhalt, in dem Sie auf das Symbol „Speichern und Schließen“ klicken.



Nachdem wir nun unseren ersten Seiteninhalt angelegt haben, werden wir vielleicht enttäuscht sein, dass im Frontend dieser Inhalt noch nicht angezeigt wird. Auf der anderen Seite währe dieses aber auch überraschend, da wir Typo3 noch nicht gesagt haben, dass überhaupt Inhalt angezeigt werden soll.

Weitere Informationen als auch eine andere Herangehensweise zum Anlegen von Seiteninhalten finden Sie im Kapitel 4.3.11.3

3.8.2 Objekt CONTENT verwenden

Damit wir Inhalte angezeigt bekommen, müssen wir dies Typo3 mitteilen. Auch müssen wir Typo3 mitteilen, aus welcher Datenbanktabelle die Inhalte kommen sollen. Seiteninhalte werden defaultmäßig in der Tabelle „tt_content“ abgelegt.

Beispiel 1

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = CONTENT
04     seite.10.table = tt_content
```

Auch wenn wir uns das Ergebnis im Frontend betrachten möchten, werden wir keinen Erfolg haben. Die Webseite bleibt leer – obwohl Inhalte angelegt sind. Der Grund hierfür: Typo3 weiß nicht, wie Inhalte überhaupt dargestellt werden sollen. Dies müssen wir Typo3 zunächst noch mitteilen.

Bei der praktischen Arbeit mit Typo3 werden sogenannte „statische Templates“ inkludiert. Nähere Informationen hierzu in den Kapiteln 2.3.3 sowie 4.3.11.1. Die folgenden Abschnitt zeigen den theoretischen Hintergrund auf.

3.8.2.1 tt_content

Bisher haben wir beim dynamischen Auslesen von Datenbankfeldern nur die Felder der Tabelle „pages“ angesprochen. Jetzt müssen wir Typo3 beibringen, wie Datensätzen der Tabelle tt_content dargestellt werden sollen. Dies geschieht auf höchster Ebene in TypoScript.

Beispiel 2

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = CONTENT
04     seite.10.table = tt_content
05
06     tt_content = TEXT
07     tt_content.field = header
```

Jedes mal, wenn die Schleife mit gefundenen Datensätzen durchlaufen wird, wird der Abschnitt “tt_content” (Zeilen 6+7) ausgeführt, um den Inhalt darzustellen. In unserem Beispiel wird somit nur die Überschrift (Datenbankfeld „header“) dargestellt.

Dies wird uns aber in der Regel nicht genügen. Zur Darstellung sind daher wesendlich komplexere Definitionen notwendig, die hier in Form von einigen Beispielen demonstriert werden sollen.

Beispiel 3

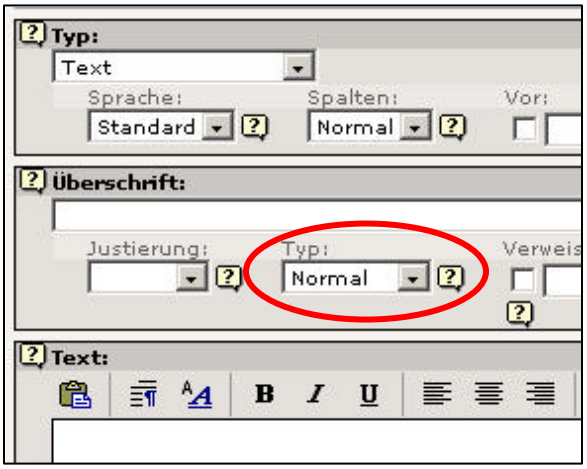
Die Überschrift soll um den Text erweitert werden. Ebenfalls soll die Überschrift in fetter Schrift dargestellt werden.

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = CONTENT
04     seite.10.table = tt_content
05
06     tt_content = COA
07     tt_content {
08        10 = TEXT
09        10.field = header
10        10.wrap = <b> | </b><br>
11
12        20 = TEXT
13        20.field = bodytext
14     }
```

- Durch das eingesetzte Objekt “COA” in Zeile 6 besteht nun die Möglichkeit, dass tt_content mehrere Objekte zugewiesen werden.
- An der Position 10 (Zeile 8) wird die Überschrift ausgegeben (mit einem Bold-Tag und einem Return gewrapt), an der Position 20 wird der Bodytext ausgegeben.

Beispiel 4

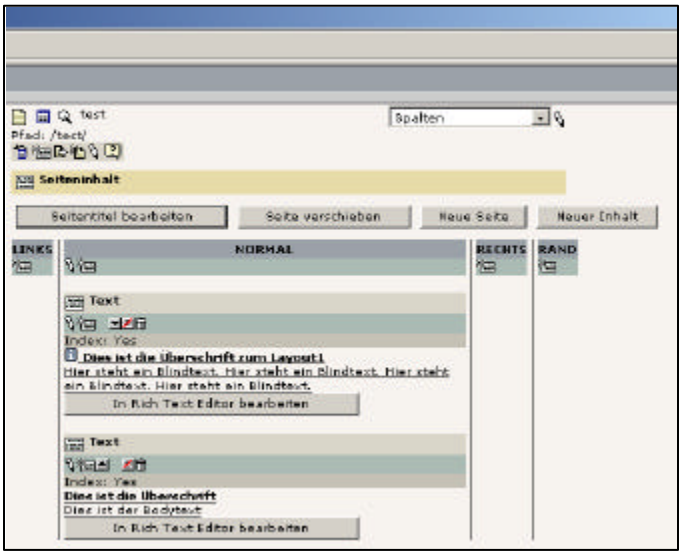
Beim Anlegen eines Seiteninhaltes kann ein Layout ausgewählt werden. Normalerweise ist dieses Feld für das Layout der Überschrift bestimmt, aber keiner hindert uns daran, dieses Layout auch für die gesamte Darstellung eines Inhaltsblocks zu nehmen.



Wir möchten in diesem Beispiel 4 zwei Seiteninhalte mit unterschiedlichen Layouts erstellen und diese auch unterschiedlich darstellen lassen.

Bei dem ersten von uns angelegten Seiteninhalt haben wir dieses Feld nicht beachtet – in der Tabelle „tt_content“ wurde in dem Datenbankfeld „header_layout“ eine „0“ (für „Normal“) gespeichert. Wenn wir bei einem neuen Datensatz „Layout1“ auswählen, wird in dem Feld eine „1“ gespeichert.

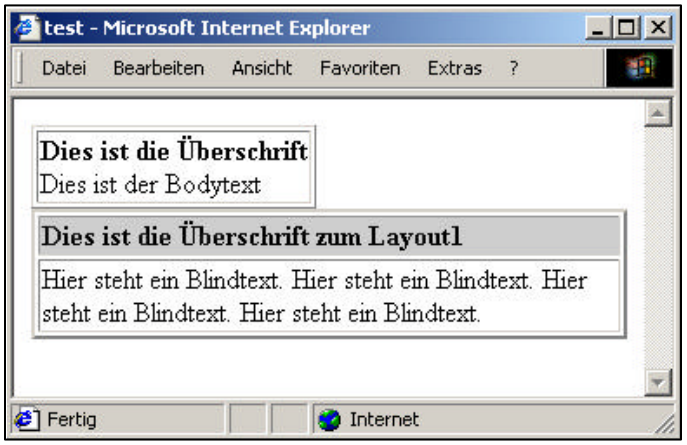
Wir legen nun also einen weiteren Seiteninhalt an – ebenfalls vom Typ „Text“, füllen das Feld Überschrift mit „Dies ist die Überschrift zum Layout1“, wählen als Layout „Layout1“ aus und als Bodytext geben wir eine beliebige Zeichenkette an.



Betrachten wir das jetzige Ergebnis vom Beispiel 3 erneut im Frontend, werden bereits beiden Datensätze angezeigt, auch wenn optisch nicht ansprechend.

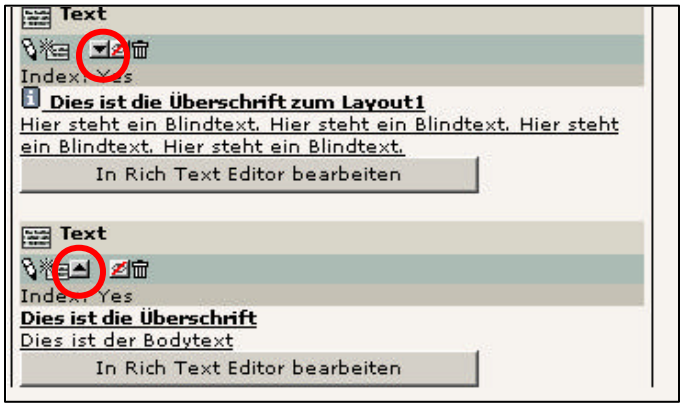
Um das gewünschte Ergebnis wie oben angegeben (unterschiedliche Layouts) zu erhalten, müssen wir eine Unterscheidung nach dem ausgewählten Layout machen.

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = CONTENT
04     seite.10.table = tt_content
05
06     tt_content = CASE
07     tt_content {
08        key.field = header_layout
09
10        1 = COA
11        1 {
12           wrap = <table border=”2”> | </table>
13           10 = TEXT
14           10.field = header
15           10.wrap = <tr><td bgcolor=”silver”><b> | </b></td></tr>
16           20 = TEXT
17           20.field = bodytext
18           20.wrap = <tr><td> | </td></tr>
19        }
20
21        default = COA
22        default {
23           wrap = <table border=”1”><tr><td> | </td></tr></table>
24           10 = TEXT
25           10.field = header
26           10.wrap = <b> | </b><br>
27           20 = TEXT
28           20.field = bodytext
29        }
30     }
```



3.8.2.2 select : sortieren

Augenscheinlich mag alles in Ordnung sein und wir sind zufrieden. Der Redakteur wird bei der praktischen Arbeit aber schnell feststellen, dass die Sortierung nicht funktioniert. Beliebige Veränderungen der Datensatzreihenfolge im Backend bleiben im Frontend ohne Erfolg. Grund hierfür ist, dass wir Typo3 noch nicht mitgeteilt haben, wie die Datensätze überhaupt sortiert werden sollen.



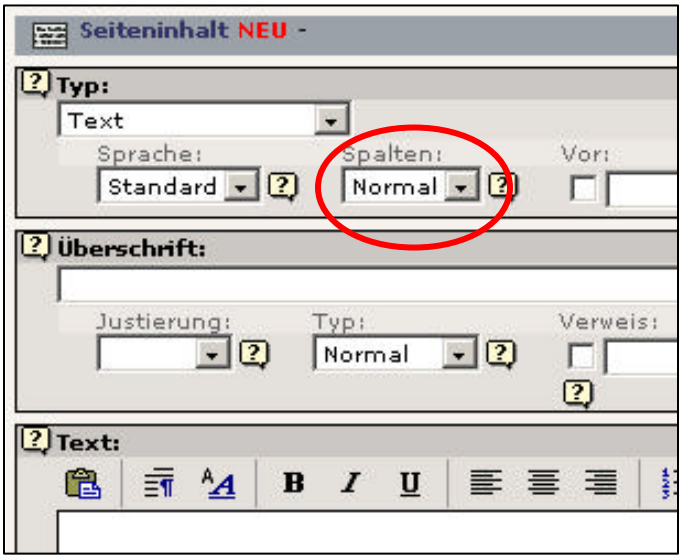
Beispiel 5

Wir erweitern unser Beispiel 4 um die Zeile 5:

```
01     [...]
03     seite.10 = CONTENT
04     seite.10.table = tt_content
05     seite.10.select.orderBy = sorting
06     [...]
```

3.8.2.3 select : Spalten

Wer ein Design verwenden möchte, dass mit unterschiedlichen Spalten arbeitet, z.B. einem normalen Inhalts-Bereich und einer rechten Spalte, wird im Backend schnell fündig. Hier kann (problemlos) angegeben werden, das ein Seiteninhalt z.B. Links, Normal, Rechts oder am Rand stehen soll.



Das Backend speichert den Wert wieder in der Datenbank in dem Datenbankfeld colPos. Folgende Werte werden gespeichert: 0 = Normal, 1 = Links, 2 = Rechts, 3 = Rand.

Um ein Ergebnis zu erreichen, dass nur Datensätze der Spalte „Normal“ angezeigt werden, muss unser Template aus Beispiel 4 bzw. Beispiel 5 erneut um die CONTENT-Eigenschaft "select" erweitert werden.

```
01     [...]
03     seite.10 = CONTENT
04     seite.10.table = tt_content
05     seite.10.select.orderBy = sorting
06     seite.10.select.where = colPos = 0
07     [...]
```

3.9 IMAGE

Mit dem Objekt IMAGE können Grafiken angezeigt werden. Typo3 nimmt hierbei die gesamte HTML-Arbeit ab und erzeugt einen img-Tag.

Beispiel 1

Wir laden zunächst eine Grafik in den Ordner fileadmin hoch. Die verwendete Grafik ist die von Windows mitgelieferte jpg-Grafik (Läufer auf blauem Grund) und wurde als fileadmin/testbild.jpg abgespeichert.

Mittels TypoScript können wir nun diese Grafikdatei anzeigen lassen:

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = IMAGE
04     seite.10.file = fileadmin/testbild.jpg
```

Beispiel 2

Das IMAGE-Objekt kann aber noch weit mehr, als nur Grafiken anzeigen zu lassen. Hier können z.B. auch Größen des Bilders geändert werden, die direkt Serverseitig berechnet werden.

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = IMAGE
04     seite.10.file = fileadmin/testbild.jpg
05     seite.10.file.width = 100
```

Die gesamte Mächtigkeit der Bildverarbeitungsfunktionen kommen aber insbesondere beim GIFBUILDER-Objekt (Abschnitt 3.10) zum Vorschein.

3.10 GIFBUILDER

Das GIFBUILDER-Objekt ist als Unterobjekt des IMAGE-Objektes zu sehen. Mit dem GUIFBUILDER können zur Laufzeit dynamisch Grafiken erstellt und modifiziert werden.

Beispiel 1

```
01   seite = PAGE
02   seite.typeNum = 0
03   seite.10 = IMAGE
04   seite.10.file = GIFBUILDER
05   seite.10.file {
06       XY = 100, 100
07       backColor = red
08   }
```

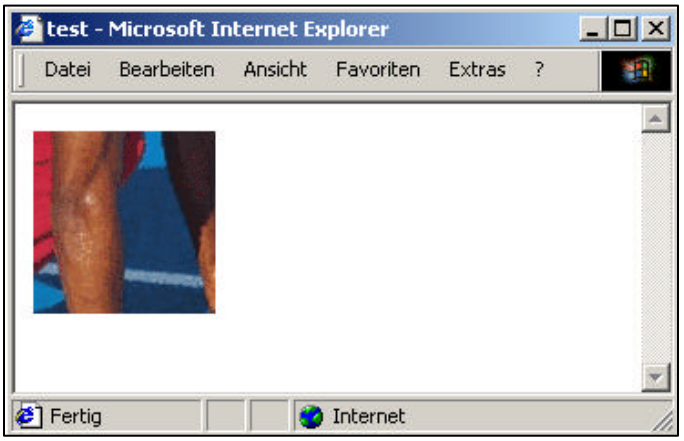
- Im Beispiel wird in Zeile 3 ein IMAGE-Objekt angelegt.
- Eine Grafikdatei ist jedoch noch nicht vorhanden, sondern wird dynamisch erstellt (Zeile 4: "file = GIFBUILDER").
- Für die Grafik wurden 2 Eigenschaften festgelegt: Die Abmaße der Grafik (Zeile 6: 100 x 100 Pixel) und die Hintergrundfarbe (Zeile 7).

3.10.1 Mit Ebenen arbeiten

Ähnlich wie bei Photoshop auch, kann der GIFBUILDER mit Ebenen umgehen, die, wie sollte es anders sein, mit 10..20..30 gekennzeichnet werden.

Beispiel 2

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = IMAGE
04     seite.10.file = GIFBUILDER
05     seite.10.file {
06        XY = 100, 100
07        backColor = red
08        10 = IMAGE
09        10.file = fileadmin/testbild.jpg
10     }
```



Wir sehen also, dass eine Ebene 10 vorhanden ist (Zeile 8+9). Diese Ebene 10 enthält eine Grafik, die unsere Hintergrundfarbe überdeckt.

3.10.2 offset : Positionieren

Mit der Eigenschaft „offset“ kann jede Ebene verschoben werden. Offset ist eine Eigenschaft der Ebene, nicht vom GIFBUILDER. Die angegebenen Werte geben die Verschiebung in Pixel nach rechts sowie nach unten an. Negative Werte sind erlaubt.

Beispiel 3

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = IMAGE
04     seite.10.file = GIFBUILDER
05     seite.10.file {
06        XY = 100, 100
07        backColor = red
08        10 = IMAGE
09        10.file = fileadmin/testbild.jpg
10        10.offset = -75, -50
11     }
```

In Zeile 10 wird die Verschiebung der Ebene 10 um 75 Pixel nach links sowie 50 Pixel nach oben angegeben.

3.10.3 Grafischer Text

Selbstverständlich sind die Funktionen des GIFBUILDERS durchaus mächtiger als die Beispiele 1 bis 3 zeigen. So kann z.B. ein Text auf eine Grafik gelegt werden. Das hierfür eingesetzte TEXT-Objekt ist jedoch nicht mit dem bereits bekannten TEXT-Objekt (Kapitel 3.2) zu verwechseln. Wir befinden uns bei dem GIFBUILDER in der grafischen Beschreibung von Elementen – HTML-Code hat an dieser Stelle nichts zu suchen. Das grafische TEXT-Objekte benötigt Eigenschaften wie Schriftgröße in Punkt, zu verwendende TTF-Datei etc.

Beispiel 4

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = IMAGE
04     seite.10.file = GIFBUILDER
05     seite.10.file {
06        XY = 200, 50
07        backColor = red
08        10 = TEXT
09        10.text = HALLO WELT
10        10.fontFile = fileadmin/fonts/verdana.ttf
11        10.fontSize = 15
12        10.fontColor = white
13        10.niceText = 1
14        10.offset = 10, 30
15     }
```

- In Zeile 6 haben wir die Dimensionen der Grafik von 100 x 100 auf 200 x 50 geändert, um einem länglichen Text gerecht zu werden.
- Der Ebene 10 wurde eine grafisches TEXT-Objekt zugewiesen (Zeile 8).
- Die Eigenschaft „text“ in Zeile 9 liefert hier einen statische Text zurück.
- Die Eigenschaften fontFile, fontSize sowie fontColor beschreiben das Aussehen des Textes. Die in fontFile angegebene ttf-Datei muss auf dem Server vorhanden sein (überprüfen Sie dieses und laden Sie eine entsprechende ttf-Datei ggf. hoch).
- In Zeile 13 wird der Kantenglätter aktiviert (weiche Kanten).

- Die Position des Textes wird ebenfalls mit offset festgelegt (Zeile 14). Beim Positionieren des Textes wird allerdings die linke, untere Kante des Textes angegeben. Diese untere Kante wird in unserem Beispiel um 10 Pixel nach rechts und 30 Pixel nach unter verschoben.

3.11.4 Ein einfacher Schatten

Beispiel 5

Wir können durch das bereits gelernte auch ohne große Umstände bereits einen einfachen Schatten realisieren. Hierzu nehmen wir uns das Beispiel 4 und erweitern dieses um einige wenige Zeilen.

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = IMAGE
04     seite.10.file = GIFBUILDER
05     seite.10.file {
06        XY = 200, 50
07        backColor = red
08        10 = TEXT
09        10.text = HALLO WELT
10        10.fontFile = fileadmin/fonts/verdana.ttf
11        10.fontSize = 15
12        10.fontColor = white
13        10.niceText = 1
14        10.offset = 10, 30
15
16        5 < .10
17        5.fontColor = black
18        5.offset = 12, 32
19     }
```

- In Zeile 16 wird die Ebene 10 in die Ebene 5 kopiert. Damit ist die Ebene 5 mit der Ebene 10 absolut identisch und liegt unterhalb von der Ebene 10.
- In den Zeile 17 und 18 nehmen wir entsprechende Modifikationen vor, um einen Schatteneffekt zu erreichen (Schwarzer Text, Positionieren um 2 Pixel weiter unten und 2 Pixel weiter rechts).

3.10.5 Mehr Dynamik

Unsere bisher erstellten Grafiken werden zwar dynamisch erstellt, die tatsächliche Dynamik fehlt jedoch. Im folgendem Beispiel wird gezeigt, wie der Text dynamisch aus der Tabelle „pages“ geholt wird und wie sich die Breite der Grafik dynamisch an die Breite des Textes anpasst. Als Grundlage wird das Beispiel 5 genommen.

Beispiel 6

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = IMAGE
04     seite.10.file = GIFBUILDER
05     seite.10.file {
06        XY = [10.w]+30, 50
07        backColor = red
08        10 = TEXT
09        10.text.field = title
10        10.fontFile = fileadmin/fonts/verdana.ttf
11        10.fontSize = 15
12        10.fontColor = white
13        10.niceText = 1
14        10.offset = 10, 30
15
16        5 < .10
17        5.fontColor = black
18        5.offset = 12, 32
19     }
```

- In Zeile 6 wurde mit [10.w]+30, 50 eine dynamische Breite definiert. [10.w] gibt an, dass die Breite der Ebene 10 genommen werden soll (tatsächliche Breite des Textes umgerechnet in Pixel).
- Die Grafik wird pauschal um 30 Pixel verbreitert, da alleine schon die der Text vom Rand 10 Pixel entfernt steht (Zeile 14).

3.11 HMENU

In diesem Kapitel und den kommenden zwei Kapiteln beschäftigen wir uns mit der Erstellung von Menüs (HMENU = hierarchisches Menü).

3.11.1 Einführung

Typo3 liefert von Hause aus die automatische Grafikerzeug mit (Kapitel 3.9 und 3.10). Dies kann man sich gerade bei den grafischen Menüs zu Eigen machen, da hier nur eine grafische Vorlage erstellt werden muss. Die „Massenproduktion“ der Grafiken mit MouseOver-Effekten etc. übernimmt Typo3 mehr oder weniger automatisch.

Warum nur „mehr oder weniger“ automatisch? Da Typo3 ein sehr mächtiges Content Management System ist, stehen einem auch und gerade bei den Menüs eine Vielzahl von Möglichkeiten zur Verfügung. Um diese Möglichkeiten nutzen zu können, wird auch hier TypoScript verwendet. Mit TypoScript lassen sich daher beliebige Menüs erstellen, die das Verhalten und Aussehen haben, das Sie gerne wünschen.

3.11.2 Vier unterschiedliche Menüarten

Es gibt 4 grundsätzlich unterschiedliche Arten von Menüs:

- Textmenüs
- Grafische Menüs
- Layer-Menüs
- JavaScript-Menüs

Die wohl einfachsten Menüs sind Textmenüs, da hier keine Grafiken im Hintergrund erzeugt werden müssen. Die restlichen drei Menüarten sind etwas anspruchsvoller, aber keineswegs schwierig zu erstellen. Der TypoScript-Sprachumfang ist hier nur umfangreicher.

3.11.3 Fünf+ Zustände von Menüelementen

Menüelemente (bzw. Menüeinträge) können fünf Zustände (und mehr) annehmen:

Zustand	Beschreibung
NO	NO beschreibt den normalen Zustand eines Menüeintrages.
RO	RollOver bzw. MouseOver. Änderungen z.B. an der Schrift bzw. an der Grafik bei einem Herüberfahren mit der Maus können hier beschrieben werden. Hinweis: Der RO-Zustand funktioniert nur richtig mit dem GMENU bzw. Menüs, die den GIFBUILDER benutzen . Andere Menüs wie z.B. das Textmenü TMENU haben RO lediglich für ggf. verwendete Grafiken, die vor oder nach dem Text angezeigt werden.
ACT	Der aktuelle Verlauf. Dies bedeutet, das die aktuelle Seite als auch alle Unterknoten der aktuellen Seite anders dargestellt werden können.
IFSUB	Wenn der Menüeintrag mindestens einen Unterknoten hat, also sich weitere Menüeinträge unterhalb eines Menüeintrages befinden, kann diese Darstellung hier anders definiert werden.
CUR	Die tatsächlich aktuelle Seite. Anders als ACT wird hier nicht der gesamte Verlauf anders definiert, sondern lediglich die wirklich aktuelle Seite.

Jeder dieser fünf Zustände kann beliebig definiert werden. Logischerweise sind die unterschiedlichen Definitionen jedoch sehr gering und dezent: Eine andere Schriftart oder ein anderer Hintergrund reichen in der Regel aus. Hierzu wird ein Zustand vollständig definiert, z.B. NO, die restlichen Zustände können durch eine Kopieranweisung kopiert und abgeändert werden (Beispiel: RO < .NO).

Praktisch lassen sich diese Zustände insbesondere bei den grafischen Menüs anwenden. Bei Text-Menüs sollte z.B. für einen RollOver-Effekt die hover-Möglichkeit von Stylesheets verwendet werden.

Diese Menüzustände werden jeweils direkt unterhalb der Menüebene angegeben.

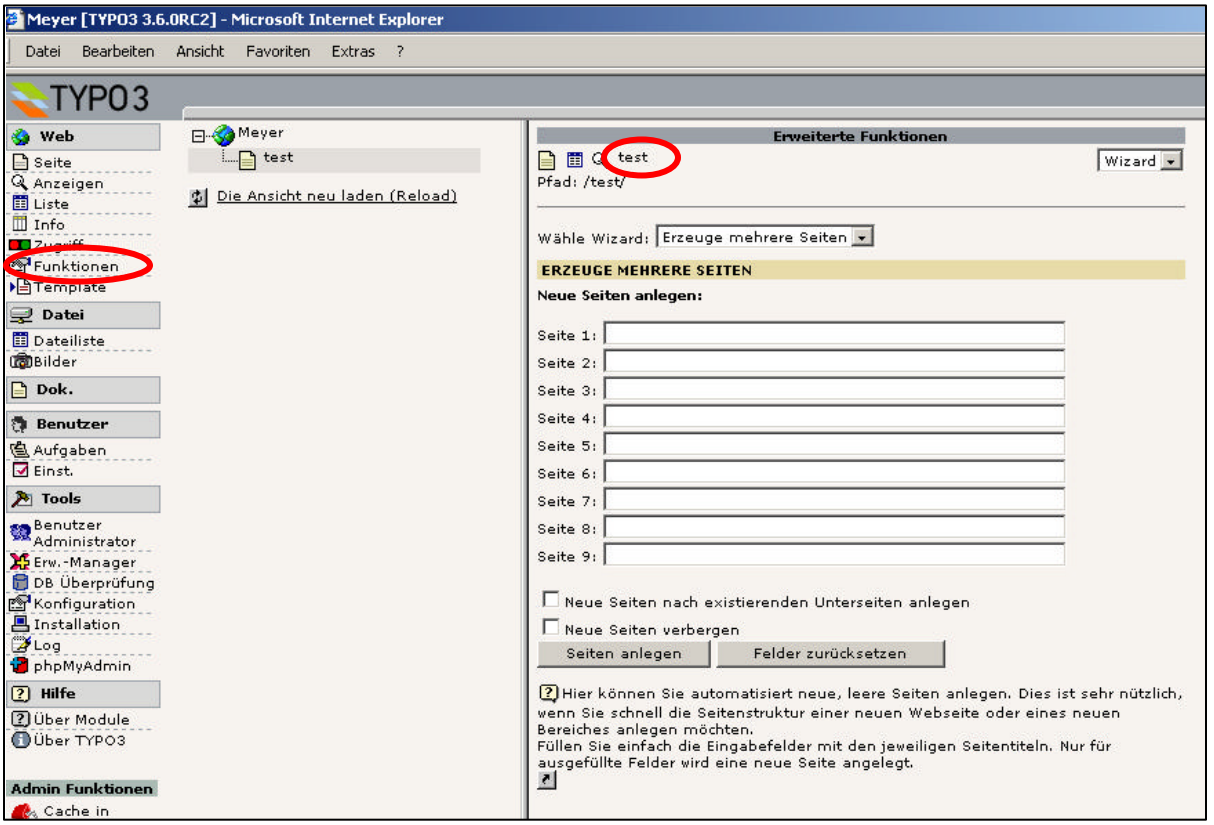
Beispiel 1 (nicht ausführbar)

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = HMENU
04     [...]
05     seite.10.1.NO = TMENU
06     [...]
07     seite.10.1.RO < seite.10.1.NO
08     seite.10.1.RO = 1
```

Ebenfalls können die Zustände untereinander kombiniert werden: ROCUR, ROIFSUB etc.

3.11.4 Vorbereitung: Seiten anlegen

Bevor wir aber überhaupt ein Menü erstellen können, werden zunächst mehrere Seiten benötigt. Diese Seiten legen wir unterhalb unserer bereits erzeugten Seite „test“ an.



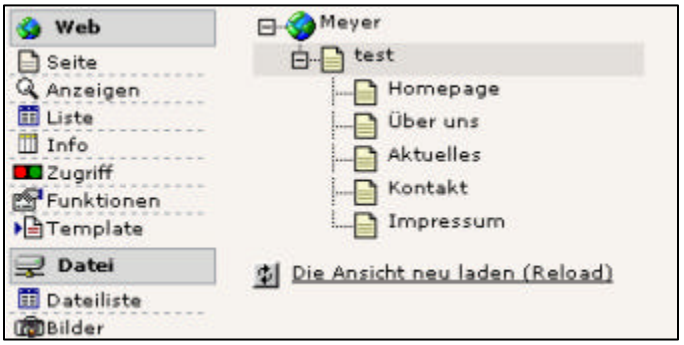
Hierzu klicken wir im linken Menü auf den Menüeintrag „Funktionen“. Wir sehen rechts eine Maske mit der Möglichkeit, bis zu 9 Unterseiten schnell anzulegen. Bevor wir nun aber Unterseiten anlegen, vergewissern wir uns, dass wir auch auf der richtigen Seite sind. Ganz oben im rechten Frame wird der Titel der aktuellen Seite angezeigt, von dem aus Unterseiten erzeugt werden.

Vergewissern kann nicht schaden, denn: Mehrere Seiten können schnell angelegt sein. Stellen wir aber fest, dass diese Seiten als Unterseiten einer falschen Seite angelegt wurden, muss jede Seite manuell wieder gelöscht werden.

Legen wir nun 5 (klassische) Seiten an:

- Homepage
- Über uns
- Aktuelles
- Kontakt
- Impressum

Das Resultat im Seitenbaum (nach Öffnen der Seite „test“ durch Anklick des Plus-Symbols) sollte nun so aussehen:



Nun haben wir die Voraussetzungen erfüllt, um ein Menü darstellen zu können.

3.11.5 special – was für ein Menü?

Mit der special-Eigenschaft kann angegeben werden, was für ein Menü erstellt werden soll. Dabei kann die Eigenschaft folgende Werte aufnehmen:

directory	Baut ein Menü ausgehend von einer angegebenen Seite auf (Unterseiten der angegebenen Seite).
rootline	Klickpfad / Brotkrumenpfad: "Sie befinden sich hier".
updated	Menü der zuletzt geänderten Seiten. Über Parameter kann z.B. das maximale Alter angegeben werden.
list	Angabe von beliebigen Seiten, aus denen ein Menü erstellt werden soll.
keywords	Mit keywords können Seiten nach "Stcihworten" angezeigt werden.

3.11.6 special : directory

Übliche Menüs werden mit dem Wert "directory" erzeugt. In unserem Fall soll z.B. ein Menü ausgehend von der Seite "test" generiert werden.

Beispiel 1 (nicht ausführbar)

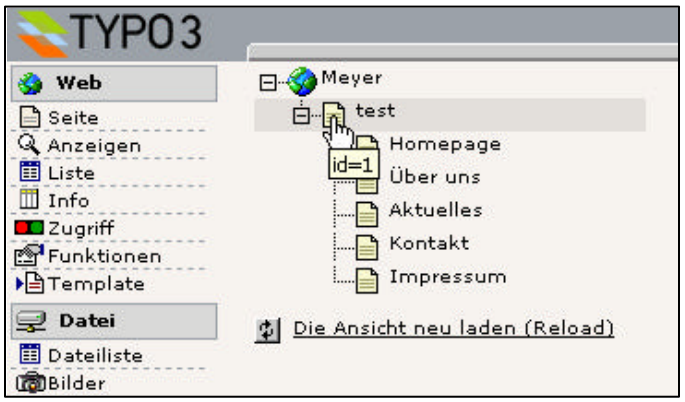
```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = HMENU
04     seite.10.special = directory
```

Dieses Beispiel ist aus zwei Gründen nicht ausführbar:

1.) Wir haben zwar angegeben, dass ein Menü ausgehend von einer Seite erzeugt werden soll, allerdings haben wir noch nicht angegeben, von welcher Seite aus das Menü erstellt werden soll
2.) Wir haben Typo3 noch nicht mitgeteilt, wie das Menü überhaupt dargestellt werden soll (Textbasiert, grafisch etc.)

Für das erste Problem können wir schnell Abhilfe schaffen: Jedes special-Menü hat bestimmte Untereigenschaften. Für special=directory müssen wir z.B. nur einen Wert angeben, von welcher Seite aus das Menü erzeugt werden soll. Dieser Wert wird mit special.value angegeben.

Als Wert müssen wir die uid (unique ID = eindeutige SeitenID) angeben, die wir im Seitenbaum erhalten. Dazu fahren wir mit der Maus über das Icon unserer übergeordneten Seite "test". Im altText wird uns die eindeutige ID der Seite angezeigt. In unserem Beispiel ist dieses die ID=1, sie kann bei Ihnen allerdings abweichen.



Diese Seiten-ID geben wir nun als Wert mit in unserem Beispiel 1 an, was aber aufgrund des unter 2.) genannten Problems noch immer nicht ausführbar ist.

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = HMENU
04     seite.10.special = directory
05     seite.10.special.value = 1
```

Um nun aber erfolgreiche Beispiele sehen zu können, wenden wir uns jetzt dem TMENU zu. In den Kapiteln 3.12 sowie 3.13 werden aber noch wesendliche Eigenschaften des HMENUs eingeführt, die sich nur an dieser Stelle mangels anschaulicher Beispiele auf eine theoretische Daseinsberechtigung berufen könnten. Daher sind diese Eigenschaften an dieser Stelle nur in der Referenz aufgeführt. Sie haben dennoch eine hohe Wichtigkeit!

3.12 TMENU

Das TMENU-Objekt ist als Unterobjekt des HMENU's zu betrachten.

Beispiel 1

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = HMENU
04     seite.10 {
05        special = directory
06        special.value = 1
07        1 = TMENU
08        1.NO = 1
09        1.NO.linkWrap = |<br>
10     }
```

Dieses Beispiel ist nun (endlich) ausführbar.

- In Zeile 3 wird das HMENU zugewiesen.
- Die Eigenschaft „special = directory“ wird in Zeile 5 gesetzt und gibt an, dass ein „klassisches“ Menü erstellt werden soll, ausgehend von einer bestimmten Seite.
- Diese bestimmte Seite wird in Zeile 6 mit „special.value“ angegeben. Der angegebene Wert ist die uid (unique-ID = eindeutige Seiten-ID) der übergeordneten Seite, bei uns also die Seite „test“ mit der uid=1.
- In Zeile 7 wird angegeben, dass die erste Menüebene ein TMENU sein soll.
- In Zeile 8 wird der Zustand NO aktiviert (optional und nicht notwendig, aber an späterer Stelle sehr nützlich).
- In Zeile 9 wird ein linkWrap definiert der angibt, wie der erzeugte <a href>-Tag gewraped werden soll. In unserem Fall also nur durch einen Zeilenumbruch.

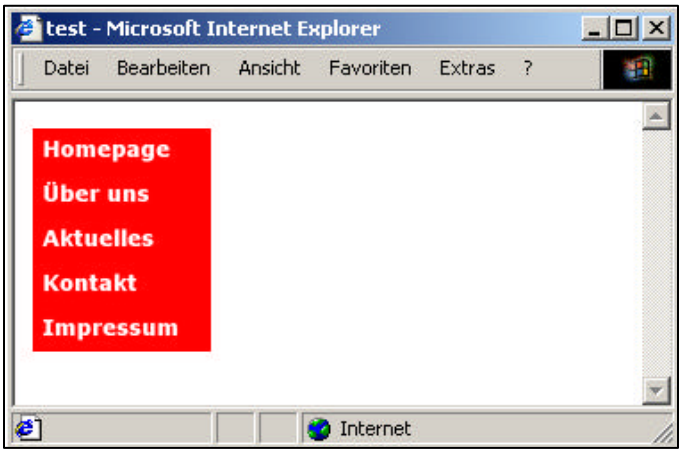
3.13 GMENU

Das GMENU ist in seiner Anwendung im Regelfall interessanter als das TMENU, da hier grafische Möglichkeiten zur Verfügung stehen, was Typo3 stark von anderen CMS-Systemen unterscheidet.

Das GMENU leitet seine Eigenschaften vom GIFBUILDER ab.

Beispiel 1

```
01     seite = PAGE
02     seite.typeNum = 0
03     seite.10 = HMENU
04     seite.10 {
05        special = directory
06        special.value = 1
07
08        1 = GMENU
09        1.NO = 1
10        1.NO {
11           XY = 100, 25
12           backColor = red
13           wrap = |<br>
14           10 = TEXT
15           10.text.field = title
16           10.fontColor = white
17           10.fontFile = fileadmin/fonts/verdanab.ttf
18           10.fontSize = 12
19           10.niceText = 1
20           10.offset = 5, 15
21        }
22     }
```



- In den Zeile 3-6 werden die allgemeinen Menüeigenschaften gesetzt, hier also dass es sich um ein klassisches Menü handeln soll (special = directory, Zeile 5) und das dieses Menü ab der Seite mit der uid 1 aufgebaut wird (special.value = 1, Zeile 6). Diese Zeilen haben wir so in identischer Form bereits beim TMENU gesehen.
- Zeile 8 gibt an, dass es sich bei der ersten Menüebene um ein GMENU handelt.
- In Zeile 9 wird der Zustand NO (=Normal) aktiviert. Dieses Aktivieren ist optional. Da allerdings andere Zustände aktiviert werden müssen, empfiehlt es sich, auch den Zustand NO zu aktivieren. Bei späteren Kopien des Zustandes NO in z.B. den Zustand RO kann dann eine explizite Aktivierung entfallen, da die Wertzuweisung = 1 ebenfalls kopiert wird.
- In den Zeilen 11-20 finden sich die Eigenschaften des GIFBUILDERS wieder (Kapitel 3.10). Hier wird die optische Erscheinung eines jeden Menüeintrages definiert.
- In der Zeile 13 wird der wrap mit „wrap = |
“ definiert. Hierdurch wird erreicht, dass die Menüeinträge untereinander stehen. Häufig wird diese Zuweisung bei Tabellenlayouts vergessen, da der Umbruch von den meisten Browsern automatisch vorgenommen wird. Die Betonung liegt allerdings bei „den meisten Browsern“.

3.13.1 Zustände einsetzen

Beim GMENU bietet es sich oftmals an, den Rollover-Zustand (bzw. MouseOver-Zustand) zu verwenden. In der Regel ist dieser Zustand fast identisch mit dem des NO-Zustandes – eine „Kleinigkeit“ wie z.B. die Hintergrundfarbe oder aber die Textfarbe wird sich lediglich ändern. Es empfiehlt sich daher, den Zustand NO als Basis zu verwenden und zu kopieren – Eigenschaften wie z.B. die Hintergrundfarbe werden anschließend überschrieben.

Beispiel 2

Dieses Beispiel erweitert das Beispiel 1 um einen RollOver-Zustand.

```
07  [...]
08      1 = GMENU
09      1.NO = 1
10      1.NO {
11         XY = 100, 25
12         backColor = red
13         wrap = |<br>
14         10 = TEXT
15         10.text.field = title
16         10.fontColor = white
17         10.fontFile = fileadmin/fonts/verdanab.ttf
18         10.fontSize = 12
19         10.niceText = 1
20         10.offset = 5, 15
21      }
22      1.RO < .1.NO
23      1.RO.backColor = green
```

Das war es prinzipiell auch schon, um einen Rollover-Effekt einzusetzen. Der Zustand NO wird in den Zustand RO kopiert (Zeile 22). Um optisch einen Effekt zu erzielen, sollte mindestens ein Wert überschrieben werden. Dieses wird in Zeile 23 durch Überschreiben der Eigenschaft backColor getätigt.

Wer in Zeile 9 den Zustand NO nicht aktivieren möchte (die Aktivierung des Zustandes NO ist optional), muss dann den Zustand RO allerdings explizit aktivieren, da die Wertzuweisung = 1 nicht mit kopiert werden würde.

```
22        1.RO < .1.NO
23        1.RO = 1
24        1.RO.backColor = green
```

3.13.2 Option Split: Elemente differenzieren

Mit sogenannten OptionSplits können diverse Eigenschaften und sogar Objekte, die in einer Schleife ausgeführt werden, aufgeteilt werden. Eine Anwendung erfolgt häufig bei den Menüs (Text-Menüs als auch grafischen Menüs). Die Möglichkeiten der Aufteilung sind zwar nur beschränkt, jedoch für den praktischen Einsatz in der Regel hinreichend.

Bei einem grafischen Menü sollen sich z.B. der erste und der letzte Menüeintrag von denen in der Mitte Unterscheiden. Die Unterscheidung soll durch eine andere Hintergrundfarbe vorgenommen werden. Das erste und das letzte Element sollen einen roten Hintergrund erhalten, die Menüelemente in der Mitte eine grüne Hintergrundfarbe.

Ein solches Vorhaben wird mit optionSplits realisiert und teilt eine Wertzuweisung in Anfang, Mitte und Ende auf.

```
hintergrundfarbe = [anfang] |*| [mitte] |*| [ende]
backColor = red |*| green |*| red
```

Beispiel 3

Dieses Beispiel erweitert das Beispiel 1 um eine OptionSplit-Möglichkeit.

```
07     [...]
08        1 = GMENU
09        1.NO = 1
10        1.NO {
11           XY = 100, 25
12           backColor = red |*| green |*| red
13           wrap = |<br>
14           10 = TEXT
15           10.text.field = title
16           10.fontColor = white
17           10.fontFile = fileadmin/fonts/verdanab.ttf
18           10.fontSize = 12
19           10.niceText = 1
20           10.offset = 5, 15
21        }
```

Das Symbol |*| unterteilt nicht nur in Anfang, Mitte und Ende, sondern auch in Erstes, Mittlere und Letztes Element. Die Unterteilung ist noch weiter möglich: Jeder Teilbereich in sich kann wieder aus erstem, zweitem etc. Element bestehen. Diese Unterteilung wird durch das Symbol | | erreicht.

Beispiel 4

```
12     backColor = red || blue |*| green |*| red
```

Dieser Code würde das erste Element mit einer roten Hintergrundfarbe, das zweite Element mit einer blauen, alle mittleren Elemente mit einer grünen und das letzte Element mit einer roten Hintergrundfarbe erscheinen lassen.

Kapitel 4: TypoScript Praxis

4.0 Vorwort

In diesem Kapitel 4 werden die Grundlagen aus Kapitel 3 verwertet und in die Praxis umgesetzt. Hierzu wurde ein konkretes Praxisbeispiel gewählt, das möglichst viele der klassischen Aufgaben und Probleme enthält.

Als Beispiel stellen wir uns zunächst eine Internetpräsentation für die Firma „Mustermann AG“ vor. Diese Präsentation soll zunächst keine besonderen Features beinhalten: 2 Navigationsbereiche („Hilfsmenü oben“ und „Hauptmenü rechts“ mit mehreren Ebenen), einen grafischen Trailer, eine Suchfunktion, eine Druckansicht sowie einen halbwegs statischen Content-Bereich auf der rechten Seite. Der eigentliche Inhalt wird in der mittleren Spalte dargestellt.

Folgendes Design wurde von der Grafikern geliefert und soll in Typo3 umgesetzt werden:

4.1 Geliefertes Design: Struktur Anlegen

Der erste Schritt zur Umsetzung des gelieferten Designs inkl. der direkt sichtbaren Funktionalitäten ist das Anlegen einer geeigneten Struktur, die in Typo3 abgebildet wird.

1.) Welche Navigationen gibt es? [Hilfsseiten]
2.) Wie viele Ebenen wird die Navigation enthalten? [Design-Aspekt]
3.) Sind Farbkonzepte vorgesehen [Extension-Templates]
4.) Wo befindet sich die Homepage?

4.1.1 Die geeignete Navigationsstruktur

Beginnen wir mit dem ersten Punkt, den es abzubilden gilt: Welche Navigationen gibt es und wie sollen diese realisiert werden?

In unserem Beispiel werden 2 Navigationen eingesetzt: Eine Navigation im oberen Teil, die als „Hilfsnavigation“ dienen soll (Homepage, Kontakt, Sitemap, Impressum etc.). Diese Navigation ist ein reines Text-Menü und enthält keine weiteren Unterebenen.

Die zweite Navigation auf der linken Seite wird mit Mouse-Over-Effekten versehen. Dieses Menü soll in diesem Beispiel mit einem grafischen Menü umgesetzt werden – die Anzahl der Unterebenen kann variieren. Der Designer hat keine grafischen Vorgaben für die Menüelemente von Unterseiten angegeben.

Beide Navigationen können mit der special-Eigenschaft (special=directory) des HMENU-Objektes abgebildet werden.

Das Design soll sich unter keinem dieser Menübereiche ändern: Sowohl bei Menüelementen aus dem oberen Hilfsmenü als auch bei Elementen aus dem linken Menü: Das gleiche Typo3-Template wird als Grundlage verwendet und an die Unterseiten vererbt.

Zur Abbildung kann folgende Seitenstruktur verwendet werden:

Seitenstruktur

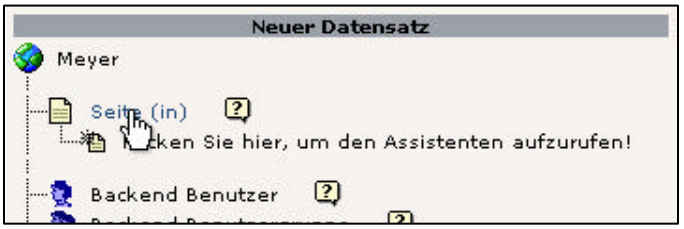
```
01     rootlevel                        Root-Ebene (Weltkugel)
02     |-- root                         Ablage des Root-Templates (Vererbung)
03     |   |-- Menü oben                Hilfsmenüpunkt für die Navigation
04     |   |   |---- Homepage           Einzelnes Menüelement
05     |   |   |---- Sitemap
06     |   |   |---- Impressum
07     |   |   \---- Kontakt
08     |   |-- Menü links               Hilfsmenüpunkt für die Navigation
09     |   |   |---- Homepage           Einzelnes Menüelement
10     |   |   |---- Über uns
11     |   |   |---- Menüitem 3
12     |   |   \---- Menüitem 4
13     |   \-- Statisch rechts          Hilfsmenüpunkt
```

4.1.2 Aufbau der Struktur im Frontend

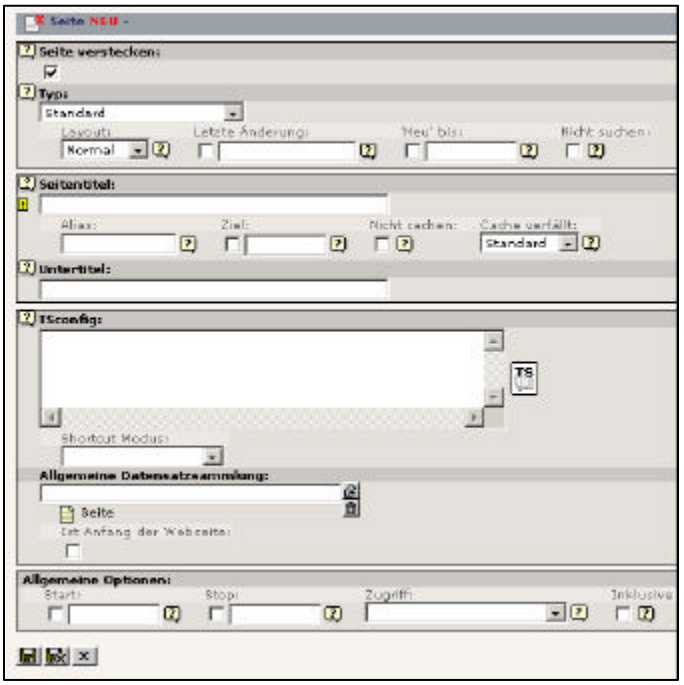
Obige Navigation soll nun in unserer Typo3-Umgebung abgebildet werden. Die Seite, die wir in Kapitel 3 bereits angelegt haben, soll bestehen bleiben, aber keinen Einfluss auf unser Projekt haben.

Hierzu legen wir auf der Weltkugel (rootlevel) eine neue Seite an, indem wir auf das Icon der Weltkugel klicken und aus dem PopUp-Menü den Eintrag „Neu“ auswählen.

Auf der rechten Seite erscheint der Dialog „Neuer Datensatz“. Hier wählen wir den Eintrag „Seite“ aus.

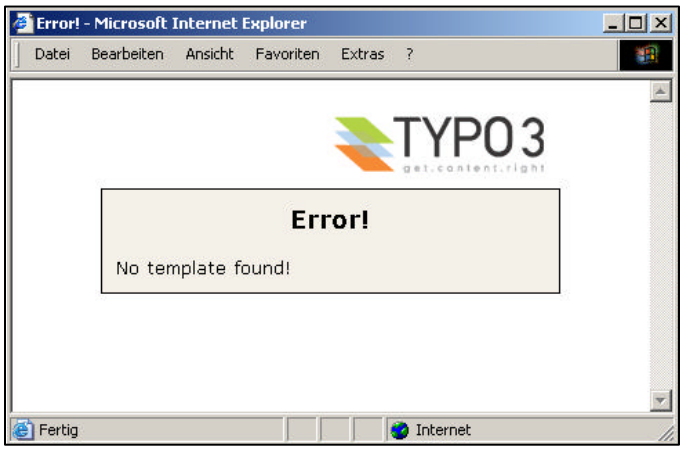


Es öffnet sich der Dialog für den „Seiten-Header“, in dem wir grundlegende Eigenschaften für eine neue Seite angeben können. Da die Seite nicht versteckt sein soll, entfernen wir das Häkchen aus „Seite verstecken“ und geben als Seitentitel „root“ an. Über das Icon „Speichern und Schließen“ wird unsere neue Seite gespeichert und der Dialog mit den Seiten-Header-Informationen geschlossen.



Wir sehen nun in der Baumdarstellung unsere neue Seite „root“ direkt unterhalb der Weltkugel (rootlevel).

Betrachten wir nun unser (leeres) Projekt, indem wir auf die Weltkugel klicken und aus dem PopUp-Menü den Eintrag „Anzeigen“ auswählen, erhalten wir die Fehlermeldung „No template found!“.

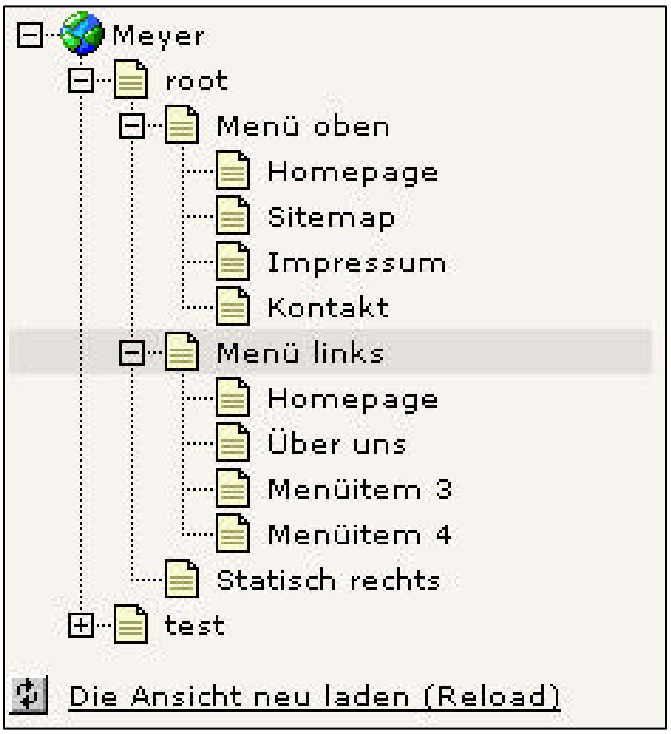


Wird im Frontend keine explizite Seiten-ID aufgerufen, z.B. mittels „http://pXXXX.typo3server.info“ oder „http://pXXXX.typo3server.info/index.php?id=0“, wird die nächstmögliche Seite unterhalb der Rootlevel (Weltkugel) ausgeführt – in unserem Fall also die Seite „root“. Um die Seite „test“ betrachten zu können, kann z.B. diese Seite durch explizite Angabe der uid (Unique ID) aufgerufen werden (http://pXXXX.typo3server.info/index.php?id=1“).

Ausgehend von der Seite „root“ legen wir nun die uns bekannten Seiten an („Menü oben“, „Menü link“, „Statisch rechts“ sowie die jeweiligen Unterseiten). Das Anlegen dieser Seiten sei hier nicht weiter erläutert und kann vom obigen Beispiel übernommen werden.

Um viele Seiten bzw. Unterseiten schnell anlegen zu können, kann das Backend-Modul „Funktionen“ genutzt werden.

Die fertige Seitenstruktur sieht wie folgt aus:

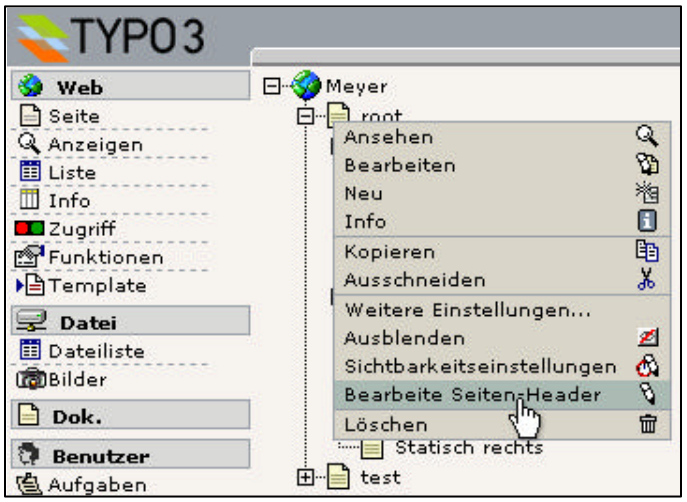


4.1.3 Hilfsseiten nicht zugänglich machen

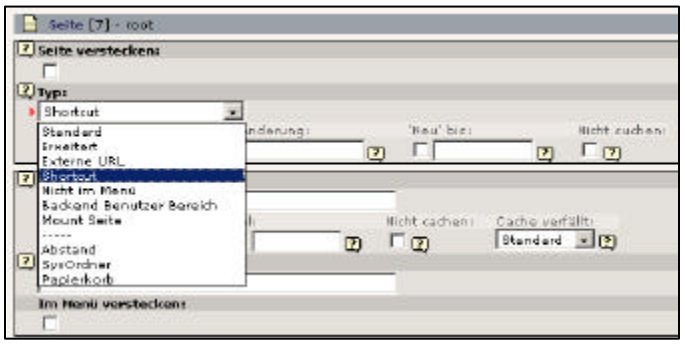
In unserer Struktur haben wir einige Seiten, die nicht zur Ablage von Inhalten bestimmt sind, dies sind im Detail die Seiten "root", "Menü oben", "Menü links" sowie "Statisch rechts". Diese Seiten gelten in unserem Projekt als "Hilfsseiten" und dienen zur Strukturierung sowie zur Template-Vererbung.

Beim Frontend-Aufruf ohne Angabe eine expliziten Seiten-ID (zum Beispiel über die Domain) würde der Besucher auf unsere Seite "root" gelangen – die aber nur existiert, um das Template zu vererben. Wir können diese Seiten aber "unzugänglich" machen: Trifft ein Besucher auf eine dieser Hilfsseiten, wird er z.B. auf die Homepage verwiesen.

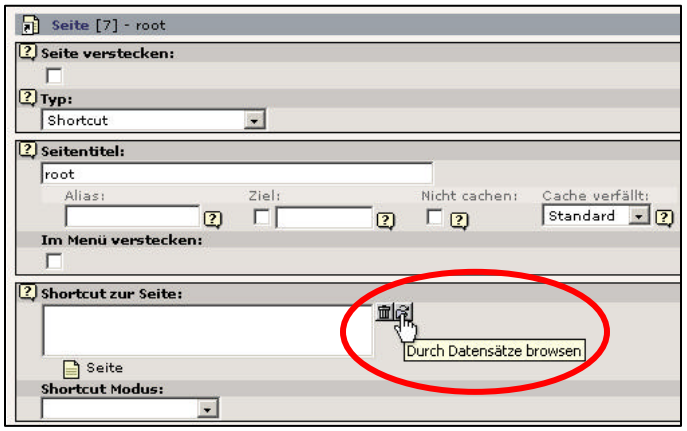
Diesen "Verweis" können wir erreichen, in dem wir den Seitentyp von "Standard" auf "Shortcut" setzen. Klicken Sie hierzu im Seitenbaum auf das Icon vor der Seite "root" und wählen Sie aus dem PopUp-Menü den Eintrag "Bearbeite Seiten-Header" aus.



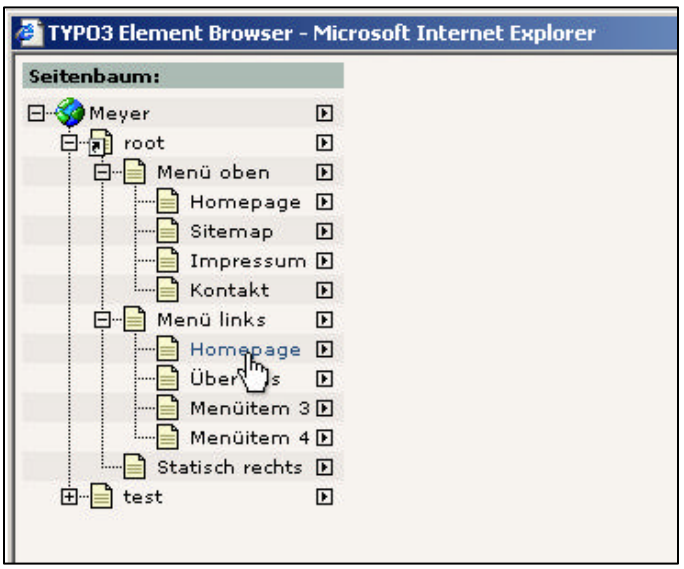
Auf der rechten Seite öffnet sich nun die Seite mit den generellen Seiten-Eigenschaften. Wählen Sie hier aus der DropDown-Box "Typ" den Eintrag "Shortcut" aus.



Bestätigen Sie die folgende Meldung mit einem "OK". Die rechte Seite mit den Header-Informationen wird neu geladen und es erscheint eine andere Maske, in der wir auswählen können, wohin der Shortcut zeigt. Klicken Sie hier auf das im Screenshot gezeigte Icon, um eine bestehende Seite auszuwählen, auf die der Shortcut zeigen soll.



Wählen Sie aus dem sich öffnendem Datensatzbrowser die Seite "Homepage" aus, die sich im Abschnitt "Menü links" befindet.



Speichern Sie danach die Seite und betrachten Sie danach die Seite im Frontend. Zum jetzigen Zeitpunkt müßten Sie eine Fehlermeldung "No template found" erhalten. Sollte jedoch eine Fehlermeldung "No propper Connection to the tree root" erscheinen, überprüfen Sie bitte, ob Sie den Shortcut richtig gesetzt haben.

Wiederholen Sie den obigen Schritt auch für die Seiten "Menü oben", "Menü links", "Statisch rechts" als auch für die Seite "Homepage", die sich im Abschnitt "Menü oben" befindet.

4.1.5 Wo befindet sich unsere Homepage?

Wir haben in unserer Seitenstruktur zwei mal den Menüeintrag "Homepage" – eine wird hiervon jedoch nur praktisch benötigt. Im obigen Abschnitt haben wir bereits festgelegt, dass unsere eigendliche Homepage die Seite sein soll, die sich im linken Menü befindet – der andere Menüpunkt in der oberen Navigation verweist lediglich auf die Homepage-Seite in der linken Navigation.

Vorteil von dieser Lösung ist z.B., dass bei einem Betrachten der Homepage der Menüeintrag auf der linken Seite auch als "aktiver Menüeintrag" dargestellt werden kann. Dennoch bekommt unsere Seite "Homepage" z.B. bei der Verwendung von Klickpfaden eine besondere Bedeutung – ein Extension-Template ist auf dieser Seite im Regelfall notwendig.

4.2 Eine Designvorlage erstellen

Auch wenn uns Typo3 das manuelle Erstellen von Webseiten abnimmt: Eine HTML-Vorlage wird dennoch benötigt, zumindest dann, wenn wir Projekte mit Designvorlagen erstellen wollen. Die Herangehensweise an Projekte mit Designvorlagen hat sich in der Vergangenheit bewährt. Die Alternative zu Designvorlagen ist, HTML-Code direkt in TypoScript zu definieren.

In diesem Kapitel arbeiten wir mehr mit unseren bewährten HTML-Tools als mit dem Typo3-Backend.

4.2.1 Präzise HTML-Ausarbeitung

Eine beispielhafte Internetseite wird, wie gewohnt, mit einem HTML-Lieblingseditor umgesetzt. Die präzise HTML-Ausarbeitung der Designvorlage legt bereits die Grundsteine für die spätere Umsetzung mit Typo3. HTML-Fehler sind zu vermeiden, da diese mit in das Typo3-Projekt aufgenommen werden würden.

4.2.2 Grafiken & Designvorlagen

Bei der Arbeit mit Designvorlagen ist bereits auf die spätere Dateistruktur zu achten. Der Aufruf einer Typo3-Präsentation findet in der Regel über http://domain.tld/index.php statt. Ausgehend von diesem Stammverzeichnis fragt der Browser die verwendeten Grafiken an. Bei Typo3-Projekten sollten wir eigene Dateien möglichst unterhalb des Ordners /fileadmin ablegen, so z.B. in /fileadmin/images. Grafiken aus unserer Designvorlage werden also in diesem Unterverzeichnis erwartet.

Um dieses Situation auf Ihrem Arbeitsplatz bei der Erstellung von Designvorlagen zu simulieren, empfiehlt es sich daher, die Grafiken ausgehend von dem Ort Ihrer Designvorlage in dem Unterordner /fileadmin/images abzulegen.

Beispiel:
Ihre HTML-Designvorlage legen Sie in einem Ordner C:\Projekte\Mustermann_AG\ ab - die verwendeten Grafiken jedoch in C:\Projekte\Mustermann_AG\fileadmin\images\.

4.2.3 Substituieren von dynamischen Elementen

Einige Bereiche unserer Internetseite bleiben statisch – andere sollen dynamisch werden.

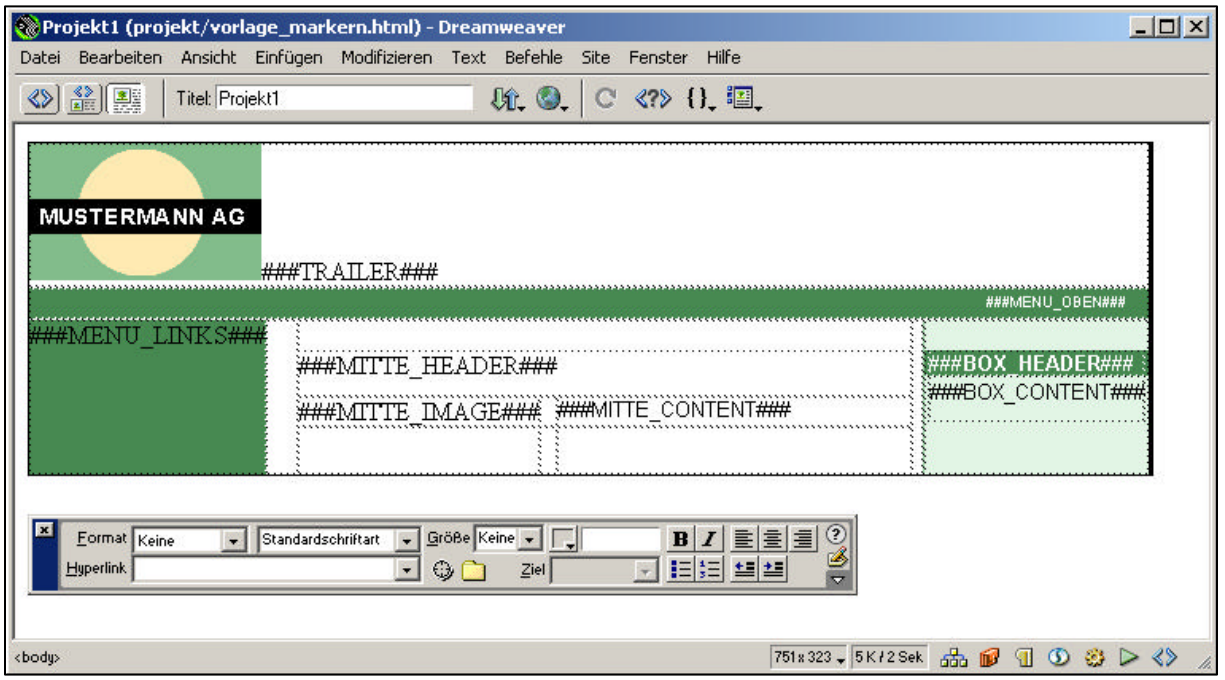
Statische Elemente unserer Präsentation:

- Die generelle Tabellenstruktur inkl. sämtlicher Hintergrundfarben und Clear-Gifs.
- Das Logo

Dynamische Elemente:

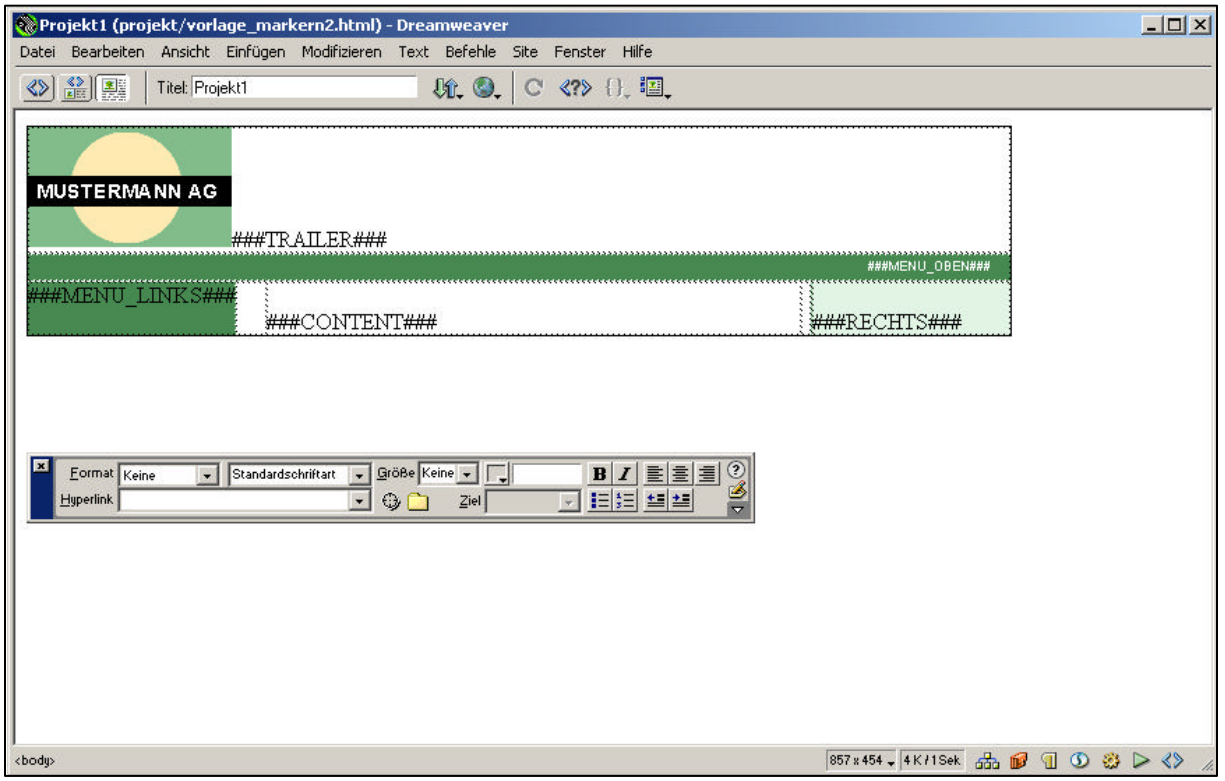
- Der Trailer „Herzlich Willkommen“: Hier soll für jede Seite ein beliebiger Text dargestellt werden können (grafisch)
- Das grafische Menü auf der linken Seite
- Das Textmenü im oberen, rechten Bereich
- Der eigentliche Content-Bereich in der Mitte
- Die News-Boxen auf der rechten Seite.

Sämtliche dynamische Elemente werden im Folgenden aus unserem erstellten HTML-Dokument entfernt und durch Platzhalter (###BEZEICHNER###) ersetzt:



Bei dem substituieren von dynamischen Bereichen ist weiterhin darauf zu achten, ob eine Verwendung z.B. eines Content-Blocks nicht ggf. mehrfach auftreten kann oder aber diese nicht ggf. sogar ganz entfallen kann.

Wir ersetzen darum auf unserer Internetseite die beiden Content-Bereiche „Mitte“ sowie „Box“ durch die Platzhalter „CONTENT“ und „RECHTS“.



Der in unserer Designvorlage verwendete HTML-Code (nicht perfekt, aber brauchbar):

```
<html>
<head>
<title>Projekt1</title>
<link rel="stylesheet" href="fileadmin/style.css" type="text/css">
</head>
<body bgcolor="#FFFFFF">

<!-- ###DOKUMENT### begin -->
<table border="0" width="706" cellspacing="0" cellpadding="0">
  <tr>
    <td bgcolor="#000000"><img src="clear.gif" width="1" height="1"></td>
    <td bgcolor="#000000"><img src="clear.gif" width="1" height="1"></td>
    <td bgcolor="#000000"><img src="clear.gif" width="1" height="1"></td>
  </tr>
  <tr>
    <td bgcolor="#000000"><img src="clear.gif" width="1" height="1"></td>
    <td width="704"><table border="0" width="100%" cellspacing="0"
            cellpadding="0">
      <tr>
        <td width="100%" valign="top" align="left"><img
            src="fileadmin/images/logo.gif"
        width="146" height="86">###TRAILER###</td>
      </tr>
      <tr>
        <td width="100%" valign="top" align="left"><img
            src="clear.gif" width="1" height="1"></td>
      </tr>
      <tr>
        <td width="100%" bgcolor="#478951" align="right"
            valign="middle"><small><img
            src="clear.gif" width="1" height="3"><br>
            <font face="Arial" size="1"
               color="#FFFFFF">###MENU_OBEN###</font><img
               src="clear.gif" width="10" height="15"
               align="middle">
       </td>
      </tr>
      <tr>
        <td width="100%" valign="top" align="left"><img
               src="clear.gif" width="1" height="1">
       </td>
      </tr>
      <tr>
        <td width="100%" valign="top" align="left"><table
            border="0" width="704" cellspacing="0"
            cellpadding="0" bgcolor="#FFFFFF">
            <tr>
               <td valign="top" align="left" width="146"
                  bgcolor="478951">###MENU_LINKS###<br>
               </td>
               <td valign="top" align="left" width="23"><img
                  src="clear.gif" width="1" height="1">
               </td>
               <td valign="top" align="left" width="386">
                  <br>###CONTENT###
               </td>
               <td valign="top" align="left" width="8"><img
                  src="clear.gif" width="1" height="1"></td>
               <td valign="top" align="left" width="1" bgcolor="#97CDA0"><img
                  src="clear.gif" width="1" height="1">
```

```
56                  </td>
57                  <td valign="top" align="left" width="140"
58                     bgcolor="#E2F5E5">
59                     <br>###RECHTS###
60               </td>
61             </tr>
62           </table>
63           </td>
64         </tr>
65       </table>
66       </td>
67       <td bgcolor="#000000"><img src="clear.gif" width="1" height="1"></td>
68     </tr>
69     <tr>
70       <td bgcolor="#000000"><img src="clear.gif" width="1" height="1"></td>
71       <td bgcolor="#000000"><img src="clear.gif" width="1" height="1"></td>
72       <td bgcolor="#000000"><img src="clear.gif" width="1" height="1"></td>
73     </tr>
74   </table>
75
76   <!-- ###DOKUMENT### end -->
77
78   </body>
79   </html>
```

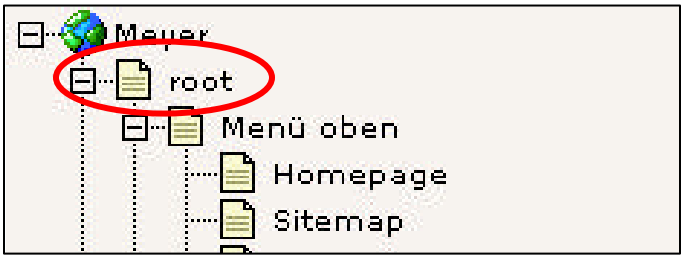
Von unserem anfänglich erstellen HTML-Dokument ist nicht mehr besonders viel übrig geblieben. Das Anfangs erstellte Dokument sollte jedoch auf allen gewünschten Browsern in der gewünschten Qualität zur Verfügung stehen. Spätere Korrekturen sind zwar immer noch möglich – in der Praxis wird die Ursache jedoch häufig in Typo3 gesucht und nicht an der Designvorlage selbst.

4.3 Umsetzung mit TypoScript

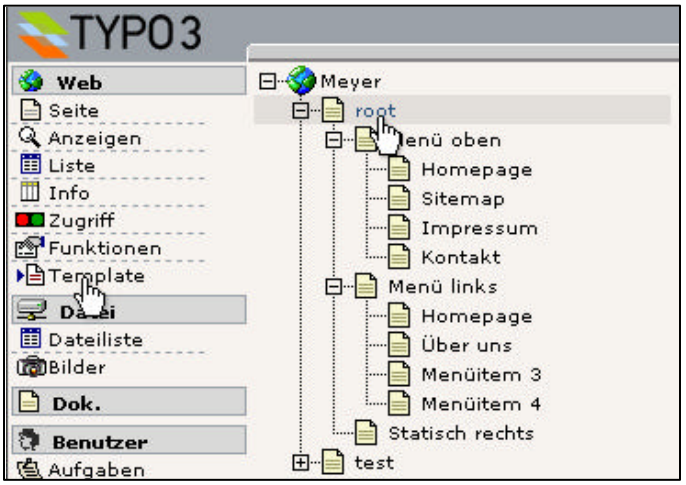
4.3.1 Das Projekttemplate erstellen

Ein Projekttemplate legen wir auf unserer Seite „Root“ an. Dieses hier angelegte Template wird vererbt und ist somit auch für die Unterseiten gültig.

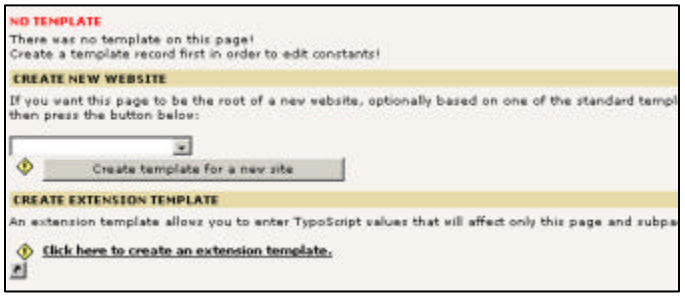
Wie gewünscht ist das in Kapitel 3 angelegte Template auf der Seite „test“ von unserem neuen Template nicht betroffen, da sich die Seite „test“ auf der gleichen Ebene befindet wie unsere Seite „root“.



Zum Anlegen eines Templates wählen wir auf der Menüliste auf der rechten Seite den Eintrag „Template“ aus aktivieren dann unsere Seite „root“ (auf die das neue Template angelegt werden soll), indem wir auf den Textlink klicken.

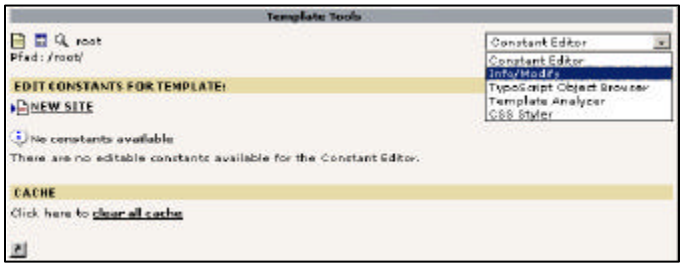


Auf der rechten Seite erhalten wir die Möglichkeit, ein Template für ein neues Projekt anzulegen, indem wir aus der Auswahlbox keinen(!) Eintrag auswählen und den Button „Create template for a new site“ anklicken.



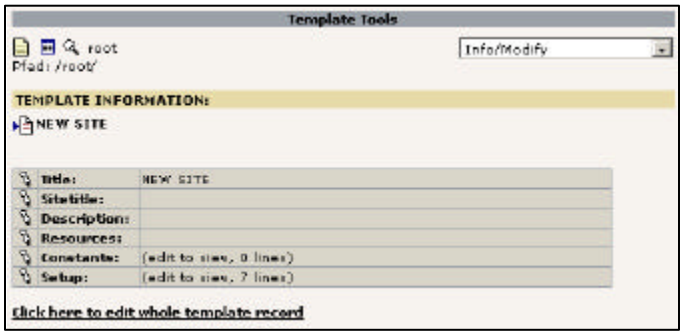
Weiterführende Informationen zu Templates erhalten Sie im Kapitel 2.

Nachdem wir nun ein neues Template erstellt haben, wählen wir in der Auswahlbox rechts oben den Menüeintrag „Info/Modify“ aus:



Wir erhalten die Übersichtsseite unseres Templates mit seinen einzelnen Bestandteilen, unter Anderem auch die Möglichkeit, im Feld „Setup“ zu editieren. Hierzu klicken wir auf das Bleistiftsystem vor dem Feld „Setup“.

In „Setup“ können wir unser zu erstellenden TypoScript-Code ablegen.



Im Feld Setup sehen wir einen beispielhaften TypoScript-Code, den wir getrost entfernen können.

4.3.2 Seiteneigenschaften festlegen

Nachdem wir nun unser Template angelegt haben, können wir beginnen, generelle Seiteneigenschaften festzulegen. Diese Eigenschaften beziehen sich im ersten Schritt auf die META-Tags sowie den Body-Tag.

Beispiel 1:

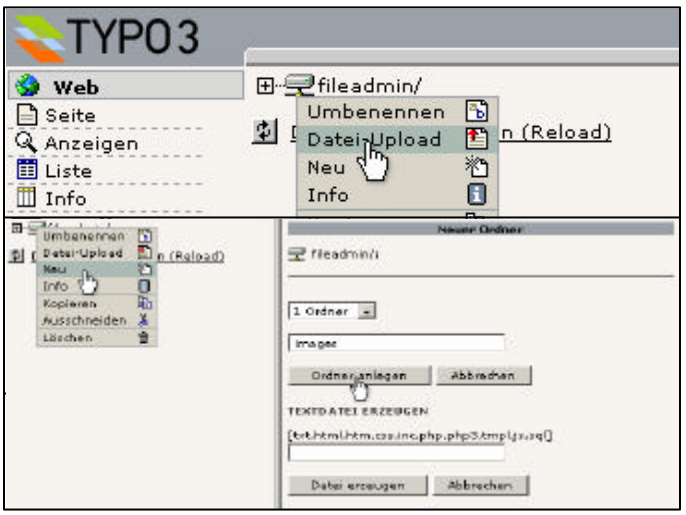
```
01     seite = PAGE
02     seite {
03        typeNum = 0
04        bodyTag = <body bgColor = "#DDDDDD">
05        stylesheet = fileadmin/style.css
06        meta.AUTHOR = Robert Meyer
07        meta.DESCRIPTION = Hier steht eine Beschreibung
08     }
```

- In Zeile 1 wird eine Instanz eines PAGE-Objektes gebildet. Der Bezeichner "seite" hat ab sofort PAGE-Eigenschaften.
- In Zeile 2 wird "seite" ausgeklammert – die Ausklammerung reicht in diesem Beispiel bis zur Zeile 8. "seite" wird somit intern bei den Zeilen 3 bis 7 vorangestellt. Somit steht in Zeile 3 statt "typeNum = 0" in Wirklichkeit "seite.typeNum = 0". Das Ausklammern erleichtert uns die Arbeit und ermöglicht eine Strukturierung. Eine bessere Übersicht ist aber nur dann gegeben, wenn durchgängig eingerückt wird.
- In Zeile 3 wird der Eigenschaft "typeNum" der Wert "0" zugewiesen. Diese Eigenschaft muss zwingend gesetzt werden. In einem Template muss es genau eins PAGE-Instanz mit typeNum = 0 geben. Nähere Informationen zur typeNum-Eigenschaft finden Sie im Kapitel 3.1
- In Zeile 4 wird der Eigenschaft "bodyTag" der gesamte HTML-Tag zugewiesen. Der HTML-Tag muss von uns explizit gesetzt werden. Wenn keine explizite Angabe erfolgt, verwendet Typo3 defaultmäßig den Wert "<body bgColor="#FFFFFF">. Auch wenn in unserem Projekt (im nächsten Abschnitt) eine Designvorlage verwendet wird: Der BodyTag befindet sich in unserer Designvorlage außerhalb des angegebenen Subparts bzw. Teilbereiches.
- In Zeile 5 wird unser Stylesheet eingelesen.
- In den Zeilen 6 und 7 werden MetaTags statisch angegeben.

Ein Betrachten dieses Templates im Frontend liefert eine leere, hellgraue Seite zurück. Ein Betrachten des HTML-Quelltextes zeigt, dass unsere Zuweisungen bereits gefruchtet haben:

```
01   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
02   <html>
03   <head>
04   <!--
05     This website is brought to you by TYPO3 - get.content.right
06   [...]
07   -->
08
09     <title>root</title>
10     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
11     <meta name="generator" content="TYPO3 3.6 CMS" />
12     <meta name="AUTHOR" content="Robert Meyer" />
13     <meta name="DESCRIPTION" content="Hier steht eine Beschreibung " />
14
15   <script language="JavaScript">
16   [...]
17   </script>
18
19   <script type="text/javascript">
20   [...]
21     </script>
22
23   </head>
24   <body bgColor = "#DDDDDD">
25
26   </body>
27   </html>
```

In Zeile 9 wird der Seitentitel unserer betrachteten Seite "root" ausgegeben. Dieses wird von Typo3 selbstständig getätigt. Durch Angabe eines "Sitetitles" bei den Template-Feldern kann der Titel um eine statische Komponente erweitert werden (z.B. "Mustermann AG: root"). Nähere Informationen hierzu finden Sie im Kapitel 2



4.3.3 Dateien mittels Dateimanager zur Verfügung stellen

Im Kapitel 4.2 haben wir unsere Designvorlage erstellt. Damit diese Designvorlage in unserer Typo3-Umgebung zur Verfügung steht, muss Sie zunächst auf unseren Server hochgeladen werden. Dieses ist prinzipiell mit FTP möglich, hier soll jedoch das Hochladen mittels dem integrierten Dateimanagers gezeigt werden. Wenn Sie die Designvorlage per FTP zur Verfügung stellen möchten, achten Sie bitte darauf, dass Sie die Designvorlage im Ordner "fileadmin" oder in einem Unterordner von "fileadmin" speichern.

Neben der Designvorlage müssen ebenfalls noch unsere verwendeten Grafiken und TTF-Schriften im System zur Verfügung gestellt werden. Die Grafiken sollen im Ordner /fileadmin/images abgelegt werden, die TTF-Schriftdateien im Ordner /fileadmin/fonts.

Zunächst müssen diese beiden Ordner angelegt werden. Um in den Dateimanager zu gelangen, klicken Sie im linken Menü auf den Menüeintrag "Dateiliste". Klicken Sie in dem sich erscheinenden Verzeichnisbaum auf das Icon vor dem Textlink "fileadmin" und wählen Sie aus dem PopUp-Menü den Eintrag "Neu" aus.

Erstellen Sie nun unterhalb von "fileadmin" zwei Ordner: "images" und "fonts".

Um nun unsere Designvorlage "vorlage.html" im System zur Verfügung zu stellen (Datei hochladen), klicken Sie im Verzeichnisbaum auf das Icon vor "fileadmin" und wählen Sie aus dem sich öffnendem PopUp-Fenster den Menüeintrag "Datei-Upload" aus. Die Designvorlage wird somit direkt im Ordner "fileadmin" abgelegt.

Wählen Sie auf der rechten Seite durch anklicken des Buttons "Durchsuchen" Ihre Designvorlage aus und laden Sie diese Datei auf den Server hoch. Laden Sie ebenfalls in den Ordner "fileadmin" das Stylesheet "style.css" hoch.

Damit Sie den aktuellen Inhalt des Ordners "fileadmin" angezeigt bekommen, klicken Sie im Verzeichnisbaum auf den Textlink des Ordners "fileadmin".

Neben der Designvorlage und dem Stylesheet müssen ebenfalls noch unsere verwendeten Grafiken und TTF-Schriften im System zur Verfügung gestellt werden. Die Grafiken legen wir im Ordner /fileadmin/images ab, die TTF-Schriftdateien im Ordner /fileadmin/fonts.

Bei den Grafiken handelt es sich in unserem Beispiel lediglich um die Datei "logo.gif". Diese laden wir, wie oben beschrieben, in den Ordner "fileadmin/images" hoch.

Als Schriftarten werden die Schriften Arial, Arial Bold, Arial Italic sowie Arial Bold Italic verwendet. Laden Sie aus Ihrem Windows/Fonts-Ordner somit die Datei arial.ttf, arialb.ttf, ariali.ttf und arialbi.ttf in den Ordner "fileadmin/fonts" hoch.

In der Regel können im "Durchsuchen"-Fenster keine TTF-Dateien mittels Doppelklick aus dem Ordner "Windows/Fonts" hochgeladen werden. Unter Windows müssen Sie hierzu den Dateinamen der TTF-Datei explizit im "Durchsuchen"-Fenster angeben.

Bei manchen Windows-Versionen kann es sein, dass für die Schriftart "Arial Bold" die Schriftdatei "arialb.ttf" mit einem großen A geschrieben wird. Vergessen Sie bitte nicht, zwecks Vermeidung von Fehlern an späteren Stellen, diese Datei so umzubenennen, dass alles klein geschrieben wird (TypoScript ist Case-Sensitive).

4.3.4 Die Designvorlage einbinden

Nachdem wir die Designvorlage nun im Ordner "fileadmin" zur Verfügung gestellt haben, können wir diese auch über TypoScript ansprechen. Hierzu erweitern wir unser Beispiel 1:

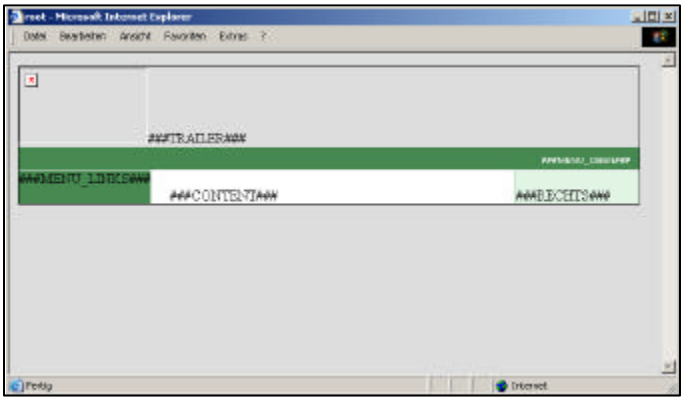
Beispiel 2:

```
01     seite = PAGE
02     seite {
03        typeNum = 0
04        bodyTag = <body bgColor = "#DDDDDD">
05        stylesheet = fileadmin/style.css
06        meta.AUTHOR = Robert Meyer
07        meta.DESCRIPTION = Hier steht eine Beschreibung
08
09        10 = TEMPLATE
10        10.template = FILE
11        10.template.file = fileadmin/vorlage.html
12     }
```

- In Zeile 9 wird "an der Position 10" eine Instanz des TEMPLATE-Objektes erzeugt.

- In Zeile 10 wird angegeben, dass die Vorlage aus einer Datei stammen soll. Der genaue Ort dieser Designvorlage wird in Zeile 11 angegeben.

Ein Betrachten des gespeicherten Templates im Frontend zeigt uns, dass die gewünschte Designvorlage geladen wird.



Bevor wir uns aber zu früh freuen: Unsere Designvorlage haben wir so angelegt, dass Subparts bzw. Teilbereiche genutzt werden. In obigem Beispiel 2 wird dieses noch nicht genutzt. Ein Blick in das erzeugte HTML-Dokument macht diesen Fehler deutlich:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
01    <html>
02    <head>
03    <!--
04      This website is brought to you by TYPO3 - get.content.right
05    [...]
06    -->
07
08    [...]
09
10    <script language="JavaScript">
11    [...]
12    </script>
13
14    </head>
15    <body bgColor = "#DDDDDD">
16    <html>
17
18    <head>
19    <title>Projekt1</title>
20    </head>
21
22    <body bgcolor="#FFFFFF">
23    <!-- ###DOKUMENT### begin -->
24    <table border="0" width="706" cellspacing="0" cellpadding="0">
```

In den Zeilen 16 bis 24 können wir den Fehler erkennen: Header- und BodyTags sind doppelt definiert. Bis einschließlich Zeile 16 wurde der HTML-Quelltext direkt von Typo3 aus erzeugt. Ab der Zeile 17 wurde unsere Designvorlage eingelesen – incl. aller dort gespeicherten HTML-Tags, wie z.B. der <head>-Abschnitt und auch der BodyTag.

In Zeile 24 können wir unseren Teilbereichskennzeichner sehen. Typo3 soll als Designvorlage nur den Abschnitt einlesen, der sich innerhalb des Teilbereiches "DOKUMENT" befindet und nicht, wie geschehen, das gesamte Dokument. Dieses kann mit Hilfe der Eigenschaft "workOnSubpart" erreicht werden.

Wir erweitern unser Beispiel 2 um folgende Zeilen:

```
01     seite = PAGE
02     seite {
          [...]
09        10 = TEMPLATE
10        10.template = FILE
11        10.template.file = fileadmin/vorlage.html
12        10.workOnSubpart = DOKUMENT
13     }
```

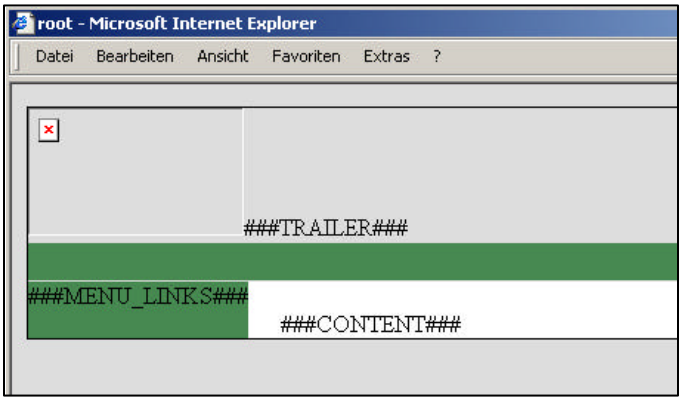
Ein erneutes Betrachten des Ergebnisses im Frontend liefert nun den gewünschten HTML-Code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
01    <html>
02    <head>
03    <!--
04      This website is brought to you by TYPO3 - get.content.right
05    [...]
06    -->
07
08    [...]
09
10    <script language="JavaScript">
11    [...]
12    </script>
13
14
15    </head>
16    <body bgColor = "#DDDDDD">
17
18    <table border="0" width="706" cellspacing="0" cellpadding="0">
```

4.3.5 Die Platzhalter ansprechen: Fehleranalyse

In diesem Abschnitt werden Tipps und Tricks zur Fehleranalyse gegeben.

Beim Betrachten der Präsentation im Frontend werden alle Verfügbaren Marker angezeigt.

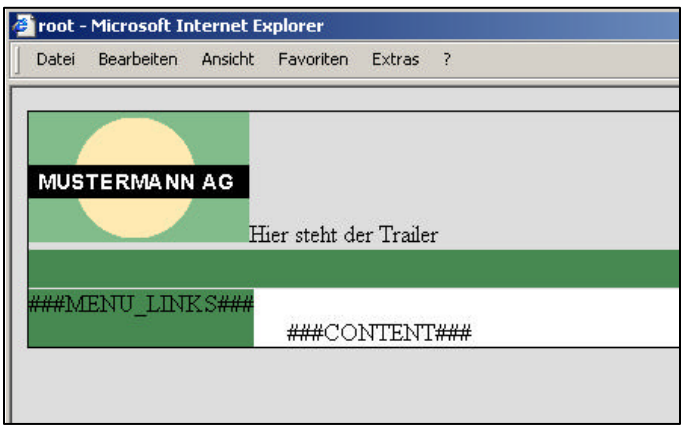


Diese sollen Schritt für Schritt ersetzt werden.

Das ersetzen eines Platzhalters geschieht mittels der Eigenschaft "marks":

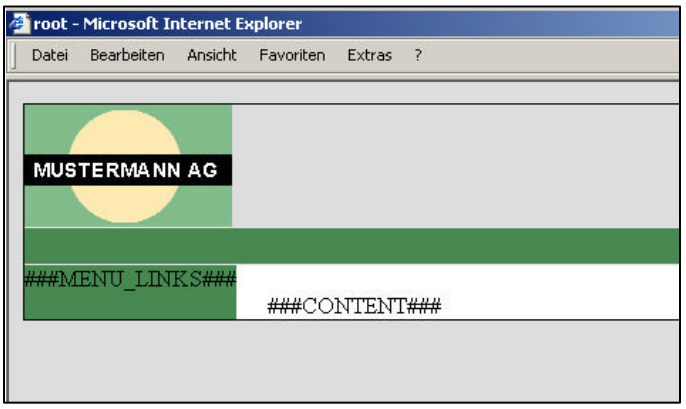
```
01     seite = PAGE
02     seite {
       [...]
09        10 = TEMPLATE
10        10.template = FILE
11        10.template.file = fileadmin/vorlage.html
12        10.workOnSubpart = DOKUMENT
13        10.marks.TRAILER = TEXT
14        10.marks.TRAILER.value = Hier steht der Trailer
15     }
```

Im Frontend können wir sehen, dass der Marker angesprochen, und auch schon das gewünschte Ergebnis zurückgeliefert wurde. Natürlich soll im Trailer im nächsten Abschnitt dynamisch eine Grafik eingelesen werden.



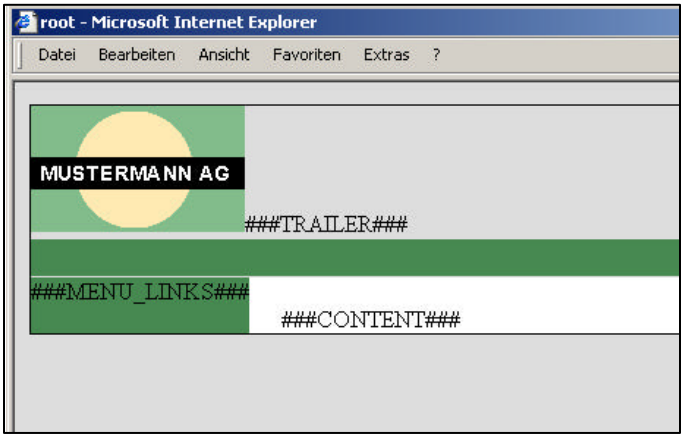
Folgender TypoScript-Code läßt jedoch den Marker bereits "verschwinden":

```
01     seite = PAGE
02     seite {
          [...]
09        10 = TEMPLATE
10        10.template = FILE
11        10.template.file = fileadmin/vorlage.html
12        10.workOnSubpart = DOKUMENT
13        10.marks.TRAILER = TEXT
14        10.marks.TRAILER.valuee = Hier steht der Trailer
15     }
```



Folgender TypoScript-Code hingegen läßt den Marker weiterhin erscheinen (TRAILER wurde falsch geschrieben):

```
01     seite = PAGE
02     seite {
          [...]
09        10 = TEMPLATE
10        10.template = FILE
11        10.template.file = fileadmin/vorlage.html
12        10.workOnSubpart = DOKUMENT
13        10.marks.TREILER =
14     }
```



Zusammenfassung:

- Der Marker wurde richtig angesprochen, wenn der Bezeichner im Frontend nicht mehr sichtbar ist (z.B. ###TRAILER###). Die Zeile 13 in obigen Beispielen bis zum Gleichheitszeichen ist somit korrekt ("10.marks.TRAILER =")

- Ist der Bezeichner im Frontend noch sichtbar, wurde der Marker nicht korrekt angesprochen. Der Fehler ist **vor** dem Gleichheitszeichen in Zeile 13 zu suchen (zum Beispiel "10.marks.TR**E**ILER =" statt "10.marks.TR**A**ILER ").

- Wird der Marker im Frontend zwar ersetzt, das gewünschte Ergebnis aber nicht zurückgeliefert, so wurde der Marker zwar korrekt angesprochen, bei der Instanzbildung oder der Zuweisung von Werten zu den Objekteigenschaften ist aber ein Fehler aufgetreten.

4.3.7 Den Trailer erzeugen

Der Trailer soll eine Grafik sein und abhängig von der aktuell aufgerufenen Seite angezeigt werden. Hierzu könnten wir für jede Seite statisch auf herkömmlichem Wege eine Grafik erzeugen und diese dann einbinden. Redakteure haben jedoch auch die Möglichkeit, neue Seiten anzulegen. Hierzu müßte dann jedes Mal eine neue Grafik erzeugt werden.

Typo3 bietet jedoch die Möglichkeit, dass Grafiken dynamisch erstellt werden. Diese Funktionalität möchten wir uns jetzt beim Trailer zu Eigen machen.

Woraus besteht der Trailer, den wir vom Grafiker geliefert bekommen haben?

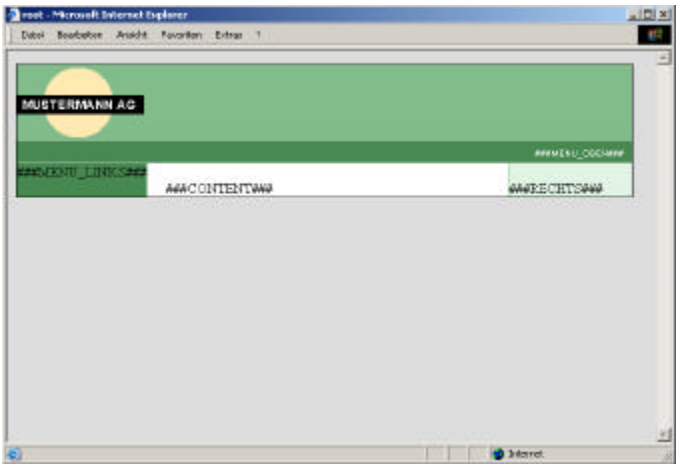
- Eine grüne Hintergrundfläche (#82BC8B) mit definierten Abmaßen (559 x 86 Pixel)
- Eine große, kursive Überschrift (ca. 60 Punkt), die auf der Hintergrundebene liegt. Schriftfarbe: #96CC9F
- Eine etwas kleinere Überschrift (ca. 36 Punkt), die mitten auf der obersten Ebene liegt. Schriftfarbe: #E2F5E5

Dieses kann direkt in TypoScript abgebildet werden.

Im ersten Schritt kümmern wir uns zunächst darum, dass eine grüne Grafik mit entsprechenden Abmaßen dargestellt wird:

Beispiel 3

```
01  seite = PAGE
02  seite {
03    typeNum = 0
04    bodyTag = <body bgColor = "#DDDDDD">
05    stylesheet = fileadmin/style.css
06    meta.AUTHOR = Robert Meyer
07    meta.DESCRIPTION = Hier steht eine Beschreibung
08
09    10 = TEMPLATE
10    10.template = FILE
11    10.template.file = fileadmin/vorlage.html
12    10.workOnSubpart = DOKUMENT
13    10.marks {
14       TRAILER = IMAGE
15       TRAILER.file = GIFBUILDER
16       TRAILER.file {
17          XY = 559, 86
18          backColor = #82BC8B
19       }
20    }
21  }
```



Ein Rechtsklick im Frontend auf unsere neue, grüne Fläche zeigt, dass es sich bei der Fläche um eine Grafik handelt.

- In Zeile 13 wird 10.marks ausgeklammert. Dies sorgt für ein übersichtlicheres Template
- In Zeile 14 wird auf dem Marker "TRAILER" eine IMAGE-Instanz gebildet. Der Marker "TRAILER" hat ab sofort IMAGE-Eigenschaften wie z.B. .file etc.
- In Zeile 15 geben wir durch TRAILER.file = GIFBUILDER an, dass keine statische Datei eingelesen werden soll, sondern die Grafik dynamisch erstellt werden soll. Aus der Eigenschaft "file" wird somit eine Instanz des GIFBUILDER-Objektes.
- In Zeile 17 und 18 geben wir die grundlegende Eigenschaften Abmaße und Hintergrundfarbe der dynamischen Grafik an.

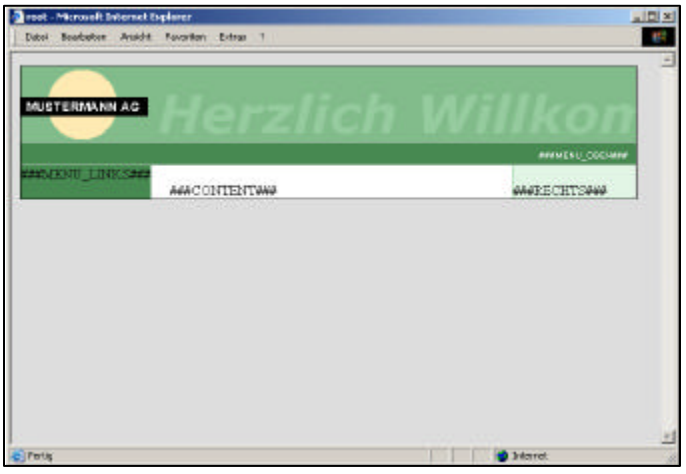
4.3.7.1 Text auf den Trailer rendern

Unser oben erstellter Trailer ist zur Zeit lediglich eine grüne Fläche. Diese soll natürlich noch um Text erweitert werden. Hierzu können wir das TEXT-Objekt des GIFBUILDERS verwenden.

Das Text-Objekt des GIFBUILDERS ist nicht zu verwechseln mit dem normalen Text-Objekt. Das grafische Text-Objekt arbeitet mit Eigenschaften wie z.B. Schriftdatei, Kantenglätter etc., die bei dem HTML-Orientieren TEXT-Objekt selbstverständlich nicht verfügbar sind.

Beispiel 4

```
01     seite = PAGE
02     seite {
          [...]
09        10 = TEMPLATE
10        10.template = FILE
11        10.template.file = fileadmin/vorlage.html
12        10.workOnSubpart = DOKUMENT
13        10.marks {
14           TRAILER = IMAGE
15           TRAILER.file = GIFBUILDER
16           TRAILER.file {
17              XY = 559, 86
18              backColor = #82BC8B
19              10 = TEXT
20              10.text = Herzlich Willkommen
21              10.fontSize = 60
22              10.fontFile = fileadmin/fonts/arialbi.ttf
23              10.fontColor = #96CC9F
24              10.niceText = 1
25              10.offset = 10, 75
26           }
27        }
28     }
```

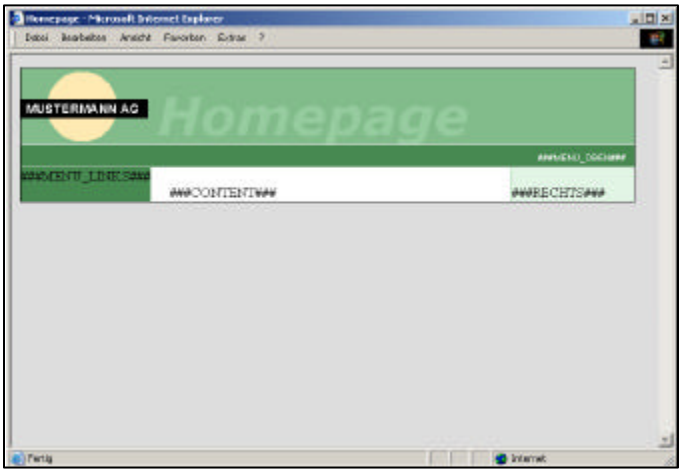


- In Zeile 19 wird auf der grafischen Ebene 10 eine grafische TEXT-Instanz erzeugt. Diese Ebenen sind vom Verständnis her mit den Photoshop-Ebenen zu vergleichen – mehrere Elemente können übereinander liegen. In unserem Fall liegt die Ebene 10 somit auf der Hintergrundebene.
- In Zeile 20 wird über die Eigenschaft "text" statisch "Herzlich Willkommen" angegeben. Dies ist somit der Text, der ausgegeben werden soll. Die Formatierung dieses Textes wird in den folgenden Eigenschaften angegeben.
- In den Zeilen 21 bis 23 wird den Eigenschaften fontFile, fontSize und fontColor jeweils ein Wert zugewiesen. Über diese Eigenschaften wird die Formatierung des Textes angegeben.
- In Zeile 24 wird durch die Eigenschaft "niceText" der Kantenglätter aktiviert.
- In Zeile 25 werden die Koordination angegeben, an denen der Text dargestellt werden soll. Die Angabe erfolgt in Pixeln ausgehend von der linken, oberen Ecke der Grafik und beschreibt die Position des Textes an der linken unteren Ecke. In unserem Fall ist die Grafik 559 Pixel breit und 86 Pixel hoch. Mit einem Offset von 10,75 wird der Text somit an der Position, ausgehend von der Grafik, 10 Pixel nach rechts und 75 Pixel nach unten dargestellt.

4.3.7.2 Den Text dynamisch darstellen

Zur Darstellung von dynamischem Text können wir die TypoScript-Funktion "field" verwenden. Beim grafischen TEXT-Objekt findet diese Funktion bei der Eigenschaft "text" Anwendung. Das Datenbankfeld in der Tabelle "pages" für den aktuellen Sitentitel lautet "title". Wir ändern unser Template wie folgt:

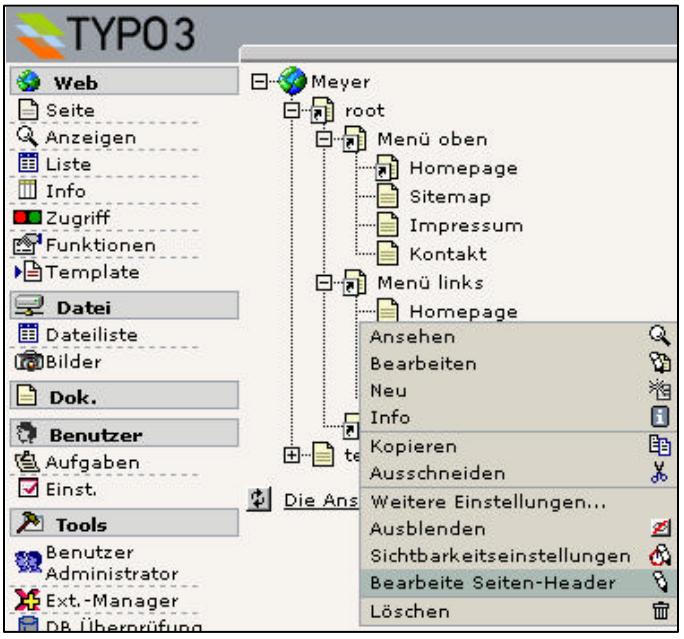
```
19          10 = TEXT
20          10.text.field = title
21          10.fontSize = 60
```



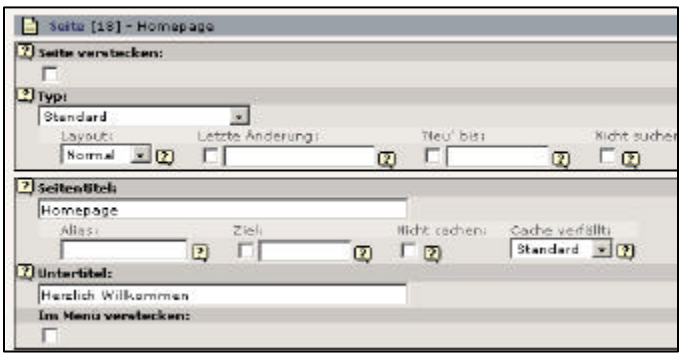
Nun soll aber auf unserer Homepage im Trailer nicht "Homepage", sondern "Herzlich Willkommen" stehen. Wir müssen es nun also ermöglichen, dass statt dem Seitentitel ein anderes Datenbankfeld ausgelesen wird. In Kapitel 3.3.1 wurde hierzu die Möglichkeit vorgestellt, mittels // mehrere Datenbankfelder abzufragen: Wenn in einem Datenbankfeld kein Wert enthalten ist, dann soll ein anderes Datenbankfeld verwendet werden.

Diese Möglichkeit können wir uns nun zu Nutze machen, indem wir das vorhandene Feld "Untertitel" (bzw. "subtitle") hierfür verwenden: Wurde ein Untertitel angegeben, so wird dieses verwendet, ansonsten der Titel.

Wir gehen also auf unsere Seite "Homepage" und bearbeiten dort den Seiten-Header für unsere Homepage:



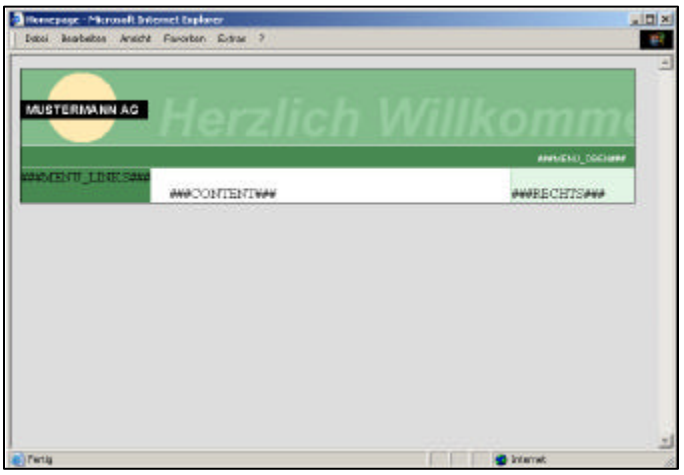
Auf der rechten Seite geben wir dann im Feld "Untertitel" unsere Trailer-Überschrift "Herzlich Willkommen" an und speichern unsere Seite.



In unserem Template müssen wir dann folgende Änderung vornehmen:

```
19          10 = TEXT
20          10.text.field = subtitle // title
21          10.fontSize = 60
```

Betrachten wir unser Ergebnis im Frontend, erscheint, wie gewünscht, unser Trailer mit einem angeschnittenem "Herzlich Willkommen".



4.3.7.3 Eine weitere Text-Ebene hinzufügen

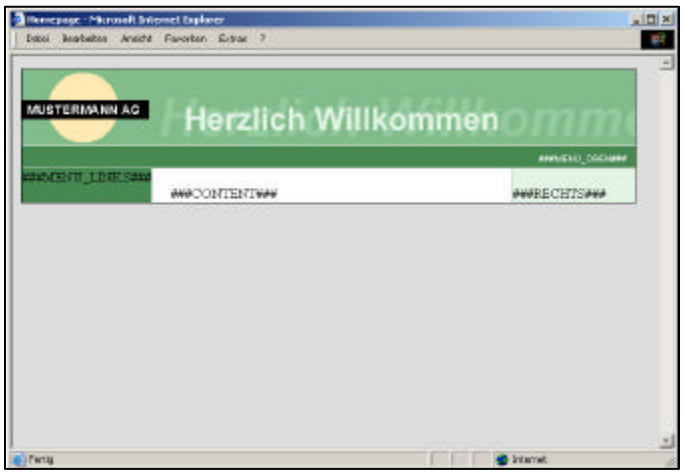
Auf unserem Trailer fehlt jedoch noch eine Textebene: Der Titel bzw. Untertitel soll ebenfalls noch auf einer höheren grafischen Ebene in einer helleren Schriftfarbe kleiner dargestellt werden.

Die genauen Eckdaten (vom Designer geliefert) lauten:

Schrift:	Arial Bold	(fileadmin/fonts/arailb.ttf)
Schriftgröße:	36 Punkt	
Schrftfarbe:	Hellgrün	(#E2F5E5)

Wir erweitern unsere dynamisch erstellte Grafik um eine weitere grafische Textebene "20":

```
01  seite = PAGE
02  seite {
      [...]
09    10 = TEMPLATE
10    10.template = FILE
11    10.template.file = fileadmin/vorlage.html
12    10.workOnSubpart = DOKUMENT
13    10.marks {
14       TRAILER = IMAGE
15       TRAILER.file = GIFBUILDER
16       TRAILER.file {
17          XY = 559, 86
18          backColor = #82BC8B
19          10 = TEXT
20          10.text.field = subtitle // title
21          10.fontSize = 60
22          10.fontFile = fileadmin/fonts/arialbi.ttf
23          10.fontColor = #96CC9F
24          10.niceText = 1
25          10.offset = 10, 75
26
27          20 = TEXT
28          20.text.field = subtitle // title
29          20.fontSize = 36
30          20.fontFile = fileadmin/fonts/arialb.ttf
31          20.fontColor = #E2F5E5
32          20.niceText = 1
33          20.offset = 40, 68
34    }
35  }
```



4.3.8 TMENU: Menü oben erstellen

Nachdem wir nun unseren Trailer fertiggestellt haben, können wir uns unserem ersten Menü im oberen Bereich widmen. Dieses Menü soll ein reines Textmenü werden – die einzelnen Menüelemente werden aus unserer Seitenstruktur entnommen.

Sprechen wir nun in TypoScript unseren Marker "MENU_OBEN" an.

```
01     seite = PAGE
02     seite {
03        typeNum = 0
04        bodyTag = <body bgColor = "#DDDDDD">
05        stylesheet = fileadmin/style.css
06        meta.AUTHOR = Robert Meyer
07        meta.DESCRIPTION = Hier steht eine Beschreibung
08
09        10 = TEMPLATE
10        10.template = FILE
11        10.template.file = fileadmin/vorlage.html
12        10.workOnSubpart = DOKUMENT
13        10.marks {
14           TRAILER = IMAGE
15           TRAILER.file = GIFBUILDER
16           TRAILER.file {
17              XY = 559, 86
18              backColor = #82BC8B
19              10 = TEXT
20              10.text.field = subtitle // title
21              10.fontSize = 60
22              10.fontFile = fileadmin/fonts/arialbi.ttf
23              10.fontColor = #96CC9F
24              10.niceText = 1
25              10.offset = 10, 75
26
27              20 = TEXT
28              20.text.field = subtitle // title
29              20.fontSize = 36
30              20.fontFile = fileadmin/fonts/arialb.ttf
31              20.fontColor = #E2F5E5
32              20.niceText = 1
33              20.offset = 40, 68
34           }
35
36           MENU_OBEN = HMENU
37        }
38     }
```

In Zeile 36 wird auf dem Marker "MENU_OBEN" eine Instanz des HMENU-Objektes gebildet. Ein Blick in das FrontEnd sollte zeigen, dass der Marker bereits angesprochen wurde.

Wenn der Marker nicht angesprochen wurde, überprüfen Sie bitte folgende Punkte:

- Zunächst, wie im Kapitel 4.3.6 beschrieben, entsteht der Fehler noch vor dem Gleichheitszeichen – also entweder bei der Schreibweise des Markers oder aber beim ausklammern.
- Befindet sich das "MENU_OBEN = HMENU" auf gleicher Höhe wie der Marker "TRAILER"? Wenn Sie alle öffnenden und schließenden Klammer "ausrechnen", darf vor dem "MENU_OBEN" nur ein "seite.10.marks" existieren.
- Prüfen Sie die Schreibweise: Häufige Fehler dürften sein: MENUE_OBEN oder MENÜ_OBEN

Woher soll das Menü kommen?

Bisher wurde nur der Marker angesprochen – ein Menü wurde noch nicht dargestellt. Wir müssen dem System noch mitteilen, von wo das Menü noch dargestellt werden soll. Hierfür eignet sich die special-Eigenschaft des HMENU-Objektes.

Zunächst müssen wir aber wissen, von welcher Seite aus das Menü dargestellt werden soll. Wir wissen zwar, dass dieses die Hilfsseite "Menü oben" ist – für TypoScript benötigen wir aber die unique-ID (uid) des Datensatzes. Wenn wir mit der Maus über das entsprechende Icon in der Baumdarstellung fahren, wird uns als Alt-Tag die entsprechende ID angezeigt:

Die Seiten-ID ist in unserem Projekt somit die ID=10.

Die Seiten-ID wird mit großer Wahrscheinlichkeit in Ihrem Projekt von der 10 abweichen. Wenn Sie die Seiten auf eine andere Art und Weise erstellt haben, zwischendurch eine Seite angelegt und wieder gelöscht haben etc., wird die ID für die Seite "Menü oben" vermutlich eine Andere als die 10 sein.

Unsere ausgelesene Seiten-ID können wir nun in der special-Eigenschaft anwenden:

```
01  seite = PAGE
02  seite {
      [...]
13    10.marks {
14       TRAILER = IMAGE
15       TRAILER.file = GIFBUILDER
16       TRAILER.file {
            [...]
34       }
35
36       MENU_OBEN = HMENU
37       MENU_OBEN.special = directory
38       MENU_OBEN.special.value = 10
39    }
40  }
```

Ein Betrachten im Frontend wird noch kein Resultat liefern – der Marker sollte zwar nicht mehr angezeigt werden, ein Menü selbst wird aber noch nicht dargestellt.

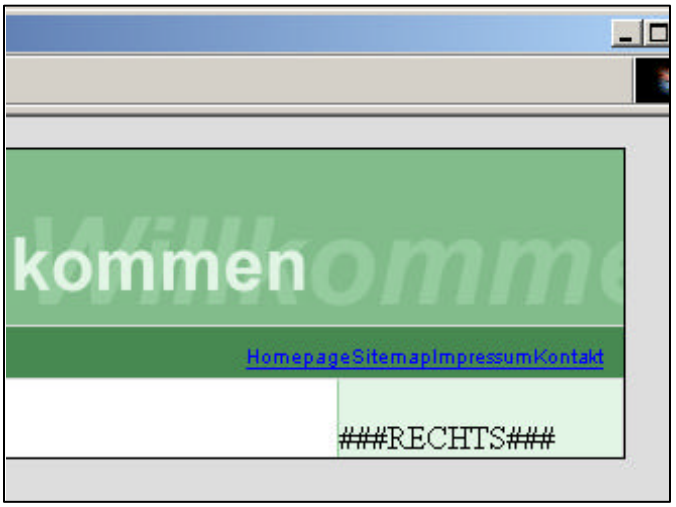
- In Zeile 36 erzeugen wir auf dem Marker "MENU_OBEN" eine Instanz des HMENU-Objektes
- In Zeile 37 geben wir über die special-Eigenschaft an, dass es bei dem erzeugten Menü um ein "diretory-Menü" handeln soll, also um ein klassisches Menü. Weitere Menü-Möglichkeiten (wie z.B. vor/zurück, Klickpfade etc. im Kapitel 3.X)
- In Zeile 38 können wir für das "special = directory"-Menü einen Wert angeben, von wo aus das Menü dargestellt werden soll. Hier wurde als Wert die "10" angegeben, da die Hilfsseite "Menü oben" die eindeutige Seite-ID "10" hat. Diese Seiten-ID wird mit großer Wahrscheinlichkeit in Ihrem Projekt abweichen! Geben Sie hier bitte Ihre Seite-ID der Seite "Menü oben" an!

Das "Menü oben" anzeigen lassen

Sorgen wir nun dafür, dass wir im Frontend auch das Menü sehen können. Hierzu müssen wir Typo3 mindestens mitteilen, wie für die erste Menüebene der normale Zustand aussehen soll.

```
01     seite = PAGE
02     seite {
          [...]
13        10.marks {
14           TRAILER = IMAGE
15           TRAILER.file = GIFBUILDER
16           TRAILER.file {
                [...]
34           }
35
36           MENU_OBEN = HMENU
37           MENU_OBEN.special = directory
38           MENU_OBEN.special.value = 10
39           MENU_OBEN.1 = TMENU
40           MENU_OBEN.1.NO = 1
41        }
42     }
```

Ein Blick in das Frontend sollte nun unser Menü erscheinen lassen:

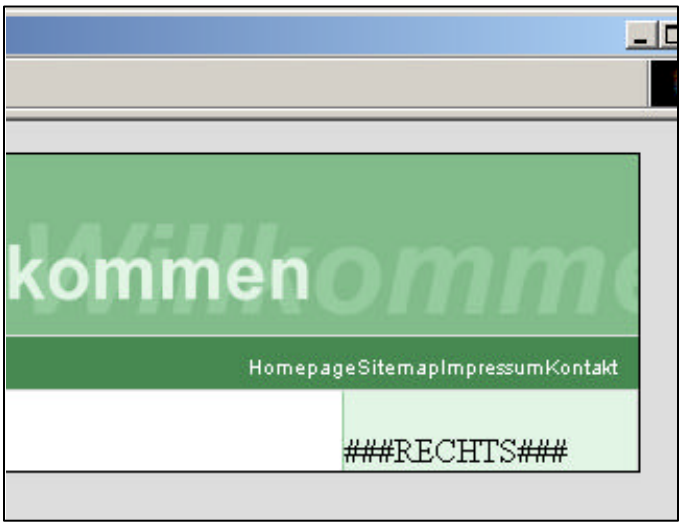


- In Zeile 39 geben wir an, dass auf der ersten Menüebene ein Textmenü (TMENU) verwendet werden soll. Es wird somit auf der ersten Ebene eine Instanz des TMENUs erzeugt und es stehen alle Eigenschaften des TMENUs zur Verfügung.
- In Zeile 40 geben wir mittels "1.NO = 1" an, dass wir den normalen Zustand aktivieren.

Stylesheet verwenden

Das Stylesheet greift, durch die Art und Weise, wie wir die Designvorlage und das Stylesheet angelegt haben, noch nicht. Wir müssen nun bei jedem Menü-Element in den A-Tag eingreifen und hier einen zusätzlichen Parameter "class = linkWeiss" angeben. Da die Links aber von Typo3 automatisch erstellt werden, stellt Typo3 eine Funktion zur Verfügung, die das Eingreifen in diesen A-Tag ermöglicht: ATagParams.

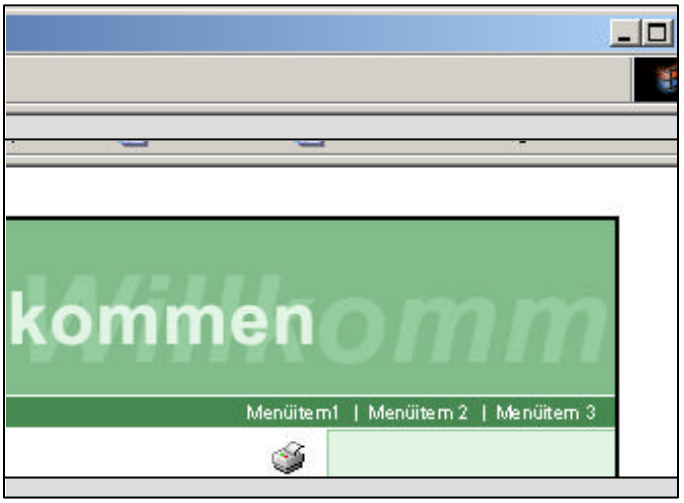
```
01        [...]
39        MENU_OBEN.1 = TMENU
40        MENU_OBEN.1.NO = 1
41        MENU_OBEN.1.ATagParams = class="linkWeiss"
```



Menüeinträge voneinander trennen

Die Menüeinträge kleben zur Zeit noch beieinander. Mit der Eigenschaft "linkWrap" können wir jeden einzelnen Menüeintrag wrappen und somit z.B. ein Leerzeichen zwischen den Einträgen mittels " " erzwingen:

```
01        [...]
39        MENU_OBEN.1 = TMENU
40        MENU_OBEN.1.NO = 1
41        MENU_OBEN.1.NO.ATagParams = class="linkWeiss"
42        MENU_OBEN.1.NO.linkWrap =  |
```



Das Ergebnis kann uns zufrieden stellen, jedoch hat der Grafiker uns eine andere Vorgabe gemacht: Zwischen jedem Menüeintrag soll ein Strich angezeigt werden.

Um dieses zu erreichen, können wir den Strich "|" mit in den linkWrap aufnehmen. Jedoch ist dieser Strich, im Folgenden als Pipe-Symbol bezeichnet, bereits für den Wrap belegt. Wir können jedoch den Ascii-Code dieses Pipe-Symbols verwenden: "|". Wir erweitern nun also unseren linkWrap um diesen zusätzlichen Strich:

```
42        MENU_OBEN.1.NO.linkWrap =  | &#124;
```

Beim linkWrap wird nun zunächst ein Leerzeichen ausgegeben, dann folgt der Menüeintrag incl. dem A-Tag, erneut eine Leerzeichen und zum Schluss unser Pipe-Symbol. Dieser linkWrap wird für jeden einzelnen Menüeintrag wiederholt, was dazu führt, dass wir immer genau ein Pipe-Symbol zuviel haben:

optionSplit für Text-Menüs mit Pipe-Symbol

Eine Lösung ist nur mittels der optionSplit-Möglichkeit gegeben. Im Kapitel 3.13.2 wurde die optionSplit-Möglichkeit bereits vorgestellt.

Nochmals zum Verständnis: Eine Menge von Einträgen kann mittels des Trennungssymbols |*| in Anfang, Mitte und Ende aufgeteilt werden: Anfang |*| Mitte |*| Ende

In der Anwendung beim linkWrap gehen wir hier wie folgt ran:

- Für den ersten Menüeintrag gilt als linkWrap: `" | &#124;"`
- Für alle Einträge in der Mitte gilt ebenfalls: `" | &#124;"`
- Für das letzte Element wird das Pipe-Symbol nicht mehr benötigt, ebenso das letzte Leerzeichen: `" |"`

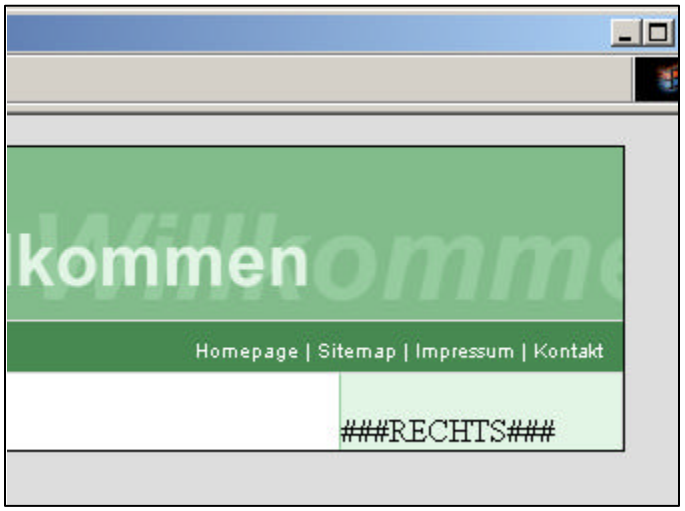
Wir setzen also zusammen: Am Anfang "` | &#124;`", in der Mitte ebenfalls "` | &#124;`" und am Ende "` |`".

Daraus folgt der (durchaus umfangreiche) linkWrap:

```
linkWrap = &amp;nbsp;|&amp;nbsp;&amp;#124; |*| &amp;nbsp;|&amp;nbsp;&amp;#124; |*| &amp;nbsp;|
```

Setzen wir dieses nun in unser Template ein und betrachten das Ergebnis im Frontend:

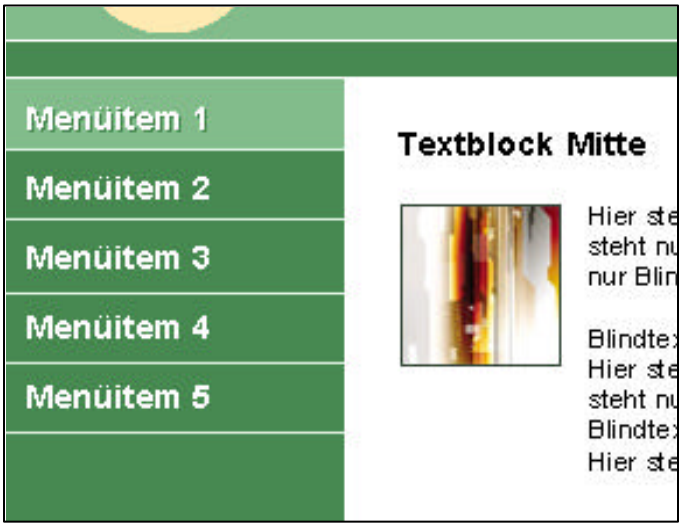
```
01     [...]
39        MENU_OBEN.1 = TMENU
40        MENU_OBEN.1.NO = 1
41        MENU_OBEN.1.NO.ATagParams = class="linkWeiss"
42        MENU_OBEN.1.NO.linkWrap = &amp;nbsp;|&amp;nbsp;&amp;#124; |*|
             &amp;nbsp;|&amp;nbsp;&amp;#124; |*| &amp;nbsp;|
```



4.3.9 GMENU: Das linke Menü erstellen

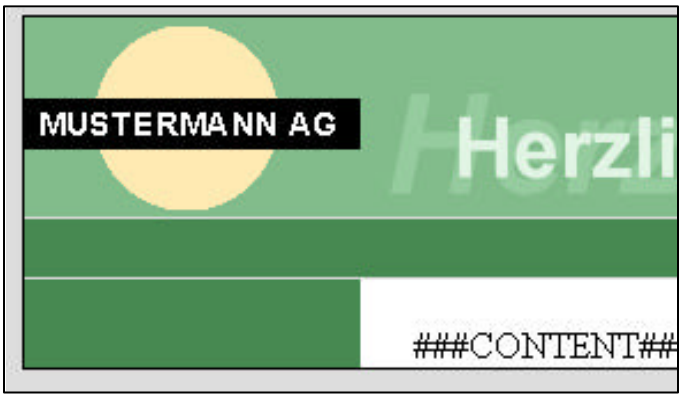
Auf der linken Seite soll, gemäß der Vorgabe der Grafiker, ein grafisches Menü dargestellt werden. Jeder Menüeintrag soll jeweils durch eine weiße Linie voneinander getrennt werden. Bei einem MouseOver als auch bei aktiven Seiten soll sich zudem die Hintergrundfarbe von ändern.

Das gelieferte Menü soll später wie folgt dargestellt werden:



Sprechen wir nun den Marker "MENU_LINKS" an. Identisch wie beim TMENU können wir auch unsere special-Eigenschaften angeben, die Seite "Menü links" hat in unserem Projekt die Seiten-ID "9" – dies kann bei Ihrem Projekt eine andere Seiten-ID sein!

```
01     seite = PAGE
02     seite {
03        typeNum = 0
04        bodyTag = <body bgColor = "#DDDDDD">
05        stylesheet = fileadmin/style.css
06        meta.AUTHOR = Robert Meyer
07        meta.DESCRIPTION = Hier steht eine Beschreibung
08
09        10 = TEMPLATE
10        10.template = FILE
11        10.template.file = fileadmin/vorlage.html
12        10.workOnSubpart = DOKUMENT
13        10.marks {
14           TRAILER = IMAGE
15           TRAILER.file = GIFBUILDER
16           TRAILER.file {
17              XY = 559, 86
18              backColor = #82BC8B
19              10 = TEXT
20              10.text.field = subtitle // title
21              10.fontSize = 60
22              10.fontFile = fileadmin/fonts/arialbi.ttf
23              10.fontColor = #96CC9F
24              10.niceText = 1
25              10.offset = 10, 75
26
27              20 = TEXT
28              20.text.field = subtitle // title
29              20.fontSize = 36
30              20.fontFile = fileadmin/fonts/arialb.ttf
31              20.fontColor = #E2F5E5
32              20.niceText = 1
33              20.offset = 40, 68
34           }
35
36           MENU_OBEN = HMENU
37           MENU_OBEN.special = directory
38           MENU_OBEN.special.value = 10
39           MENU_OBEN.1 = TMENU
40           MENU_OBEN.1.NO = 1
41           MENU_OBEN.1.NO.ATagParams = class="linkWeiss"
42           MENU_OBEN.1.NO.linkWrap =  | &#124; |*|
                | &#124; |*|  |
43
44           MENU_LINKS = HMENU
45           MENU_LINKS.special = directory
46           MENU_LINKS.special.value = 9
47        }
48     }
```



- In Zeile 44 wird auf dem Marker "MENU_LINKS" eine HMENU-Instanz erzeugt. Der Marker MENU_LINKS hat somit generelle Menü-Eigenschaften
- In Zeile 45 teilen wir Typo3 mit, dass das Menü ein "klassisches" Menü sein soll, welches von einer bestimmten Seite aufgebaut werden soll
- In Zeile 46 geben wir mittels special.value die Seiten-ID (uid) an, von der aus die Seite aufgebaut werden soll. Die Seiten-ID kann in Ihrem Projekt von der hier angegebenen Seiten-ID "9" abweichen!

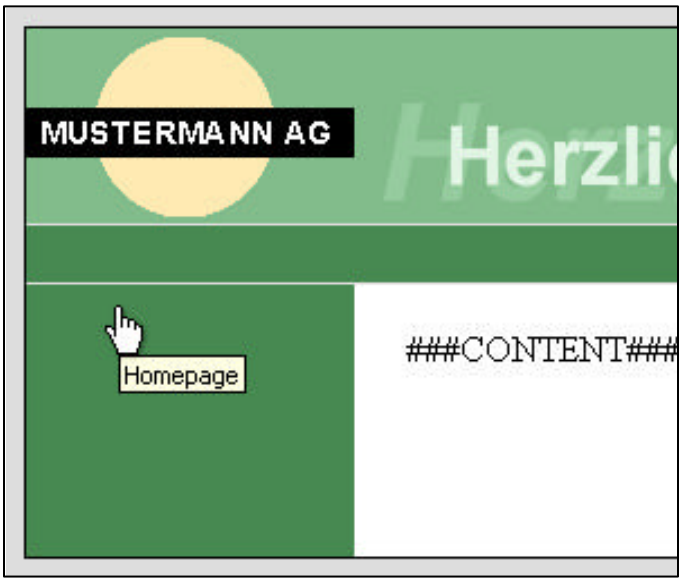
Ein Betrachten der Seite im Frontend wird kein Menü darstellen lassen – der Marker selbst sollte aber nicht mehr dargestellt werden. Falls der Marker noch dargestellt wird, finden Sie entsprechende Informationen im Kapitel 4.3.6 sowie 4.3.8.

Grafische Menüeinträge erzeugen lassen

Jeder Menüeintrag soll mit einer dunkelgrünen Hintergrundfarbe versehen werden. Ebenfalls soll, zumindest für den normalen Zustand, eine Textebene zum Anzeigen des Navigationstextes, auf die Grafik gerendert werden.

Wir erweitern unser TypoScript um die entsprechenden Angaben:

```
44        MENU_LINKS = HMENU
45        MENU_LINKS.special = directory
46        MENU_LINKS.special.value = 9
47        MENU_LINKS.1 = GMENU
48        MENU_LINKS.1.NO = 1
49        MENU_LINKS.1.NO {
50           XY = 146, 30
51           backColor = #478951
52        }
```



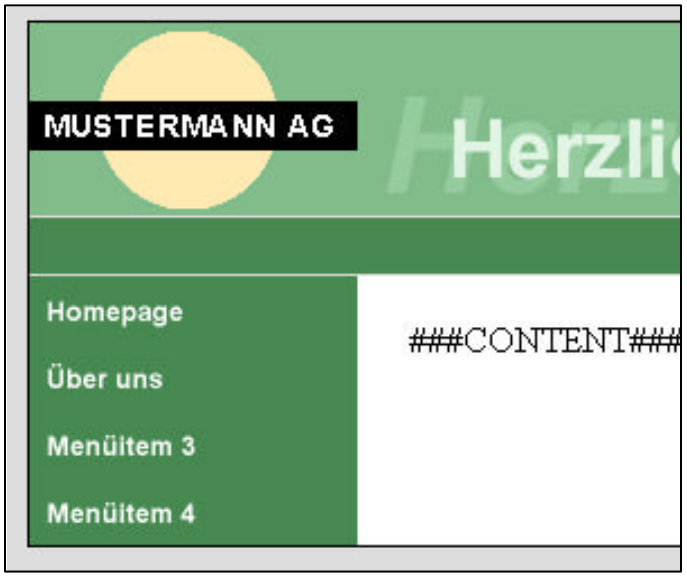
Betrachten wir das Ergebnis im Frontend, können wir erkennen, dass sich bereits etwas getan hat. Einzelne Menüpunkte sind zwar nicht direkt sichtbar – ein Mouse-Over über die grüne Fläche zeigt jedoch schon, dass sich hier Menüelemente befinden.

Text auf die Grafik rendern

Im Folgenden möchten wir nun eine grafische Textebene auf den grünen Hintergrund rendern:

```
44        MENU_LINKS = HMENU
45        MENU_LINKS.special = directory
46        MENU_LINKS.special.value = 9
47        MENU_LINKS.1 = GMENU
48        MENU_LINKS.1.NO = 1
49        MENU_LINKS.1.NO {
50           XY = 146, 30
51           backColor = #478951
52
53           10 = TEXT
54           10.text.field = title
55           10.fontColor = #FFFFFF
56           10.fontFile = fileadmin/fonts/arialb.ttf
57           10.fontSize = 12
58           10.niceText = 1
59           10.offset = 7, 21
60        }
```

Betrachten wir das Ergebnis im Frontend, werden unsere Menüeinträge bereits richtig angezeigt:



- In Zeile 53 wird als Ebene 10 eine Textebene angelegt. Diese Ebene 10 befindet sich auf der Hintergrundebene.
- In Zeile 54 wird angegeben, woher der Inhalt genommen werden soll. Hier: dynamisch aus dem Datenbankfeld "title".
- In den Zeilen 55 bis 57 werden die Eigenschaften zur verwendeten Schrift angegeben: Schriftfarbe, Schriftdatei und Schriftgröße (in Punkt).
- In Zeile 58 wird für die Schrift der Kantenglätter aktiviert. Hierdurch werden runde Kanten ermöglicht – der Text wirkt nicht mehr "pixelig".
- In Zeile 59 werden die Koordinaten beschrieben, an denen der Text angezeigt werden soll. Die Koordinaten geben den Abstand vom linken, unteren Punkt des Textes zur linken oberen Ecke der gesamten Grafik an.

Fehlende weiße Linien ergänzen

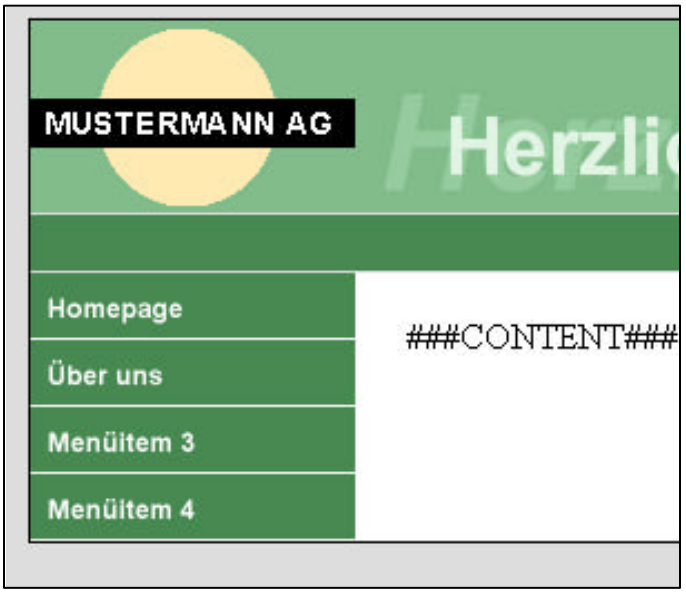
In der vom Grafiker gelieferten Vorlage werden die Menüelemente mit einer weißen Linie voneinander getrennt. Eine weiße Linie (oben) ist bereits statisch mit in die Designvorlage aufgenommen worden. Die anderen Linien müssen nun noch ergänzt werden.

Hier gäbe es zwei Möglichkeiten der Realisierung:

1. Mit einem Wrap: Nach jedem Menüelement wird eine weiße Linie mit eingewrapt. In TypoScript könnte dieses so lauten:
   wrap = |<img src="fileadmin/linie.gif" width="146" height="1">
   Dieses Beispiel ist auch anwendbar.
2. Die Linie wird direkt mit in die Grafik aufgenommen.

Anhand der zweiten Variante soll nun die weiße Linie mit in die Grafik eingearbeitet werden. Hierzu können wir zum Beispiel auf der Ebene 20 eine Linie reinladen. Diese Linie ist aber nicht vorhanden und soll dynamisch mittels dem Gifbuilder erzeugt werden:

```
44        MENU_LINKS = HMENU
45        MENU_LINKS.special = directory
46        MENU_LINKS.special.value = 9
47        MENU_LINKS.1 = GMENU
48        MENU_LINKS.1.NO = 1
49        MENU_LINKS.1.NO {
50           XY = 146, 30
51           backColor = #478951
52
53           10 = TEXT
54           10.text.field = title
55           10.fontColor = #FFFFFF
56           10.fontFile = fileadmin/fonts/arialb.ttf
57           10.fontSize = 12
58           10.niceText = 1
59           10.offset = 7, 21
60
61           20 = IMAGE
62           20.file = GIFBUILDER
63           20.file {
64              XY = 146,1
65              backColor = #FFFFFF
66           }
67           20.offset = 0, 29
68        }
```



- In Zeile 61 wird eine weitere Ebene 20 angelegt als Instanz des IMAGE-Objektes. Die hier geladene bzw. erzeugte Grafik liegt somit auf der Ebene 10 und würde diese überdecken. In unserem Beispiel mit einer 1-Pixel hohen Linie sollte eine Überdeckung kaum möglich sein.
- In Zeile 62 geben wir an, dass die Grafik dynamisch mit dem GIFBUILDER erzeugt werden soll.

- In den Zeilen 64 und 65 geben wir an, dass diese Grafik 146 Pixel lang sein soll und nur 1 Pixel hoch. Als Hintergrundfarbe wird weiß angegeben.
- In Zeile 67 verschieben wir die erzeugte Grafik um 0 Pixel nach rechts und 29 Pixel nach unten.

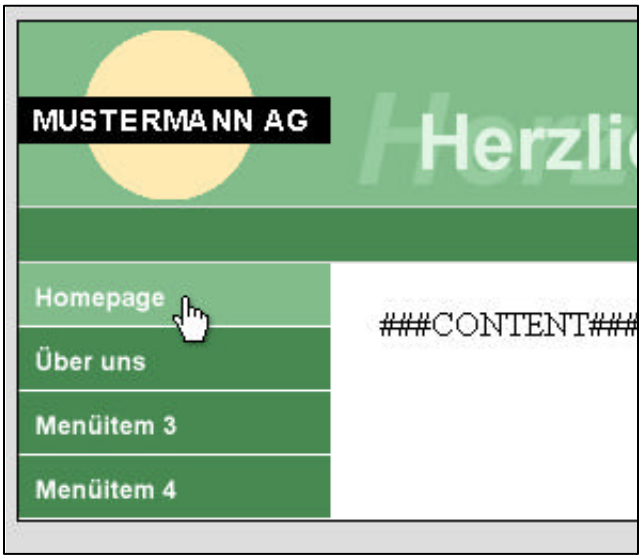
Unterschiedliche Menüzustände integrieren

Ebenfalls vom Grafiker vorgegeben ist das Layout eines Menüpunktes bei einem RollOver sowie wenn der Menüpunkt zur aktuellen Seite gehört.

Menüzustände wie RollOver weichen im Regelfall nur minimal vom normalen Zustand ab. Oft ist es nur eine dezente Veränderung – in unserem Fall soll die Hintergrundfarbe leicht heller sein und der Original-Grünton als Schatten des Textes erscheinen.

Richten wir hierzu zunächst den RollOver mit veränderter Hintergrundfarbe ein:

```
44        MENU_LINKS = HMENU
          [...]
47        MENU_LINKS.1 = GMENU
48        MENU_LINKS.1.NO = 1
49        MENU_LINKS.1.NO {
             [...]
53           10 = TEXT
             [...]
61           20 = IMAGE
             [...]
68        }
69        MENU_LINKS.1.RO < .MENU_LINKS.1.NO
70        MENU_LINKS.1.RO.backColor = #82BC8B
```

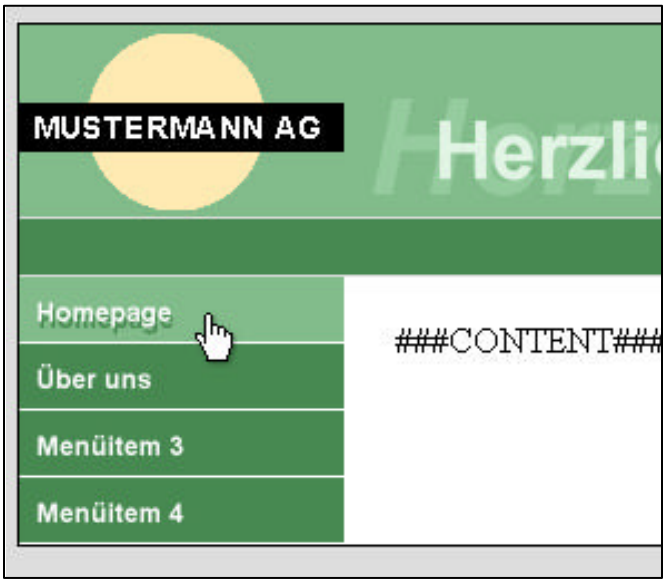


- In Zeile 69 wird mittels relativer Kopie der Zustand NO in den Zustand RO kopiert. Bei diesem Kopiervorgang werden sämtliche Eigenschaften mit kopiert und stehen als Kopie in RO zur Verfügung. Da in Zeile 48 der Zustand NO aktiviert wurde, wird durch den Kopiervorgang nun der Zustand RO ebenfalls aktiviert.
- In Zeile 69 wurde die Eigenschaft backColor, die durch den Kopiervorgang mit dem dunkleren Grünton zugewiesen wurde, überschrieben.

Eine zusätzliche Textebene hinzufügen

Für den Roll-Over-Zustand soll noch eine weitere Textebene existieren, die in der Schriftfarbe dunkelgrün um jeweils zwei pixel nach rechts und nach unten unterhalb der Ebene 10 erscheinen soll.

```
44        MENU_LINKS = HMENU
          [...]
47        MENU_LINKS.1 = GMENU
48        MENU_LINKS.1.NO = 1
49        MENU_LINKS.1.NO {
             [...]
53           10 = TEXT
             [...]
61           20 = IMAGE
             [...]
68        }
69        MENU_LINKS.1.RO < .MENU_LINKS.1.NO
70        MENU_LINKS.1.RO {
71           tmp < .backColor
72           backColor = #82BC8B
73           5 < .10
74           5.fontColor < .tmp
75           5.offset = 9,23
76        }
```



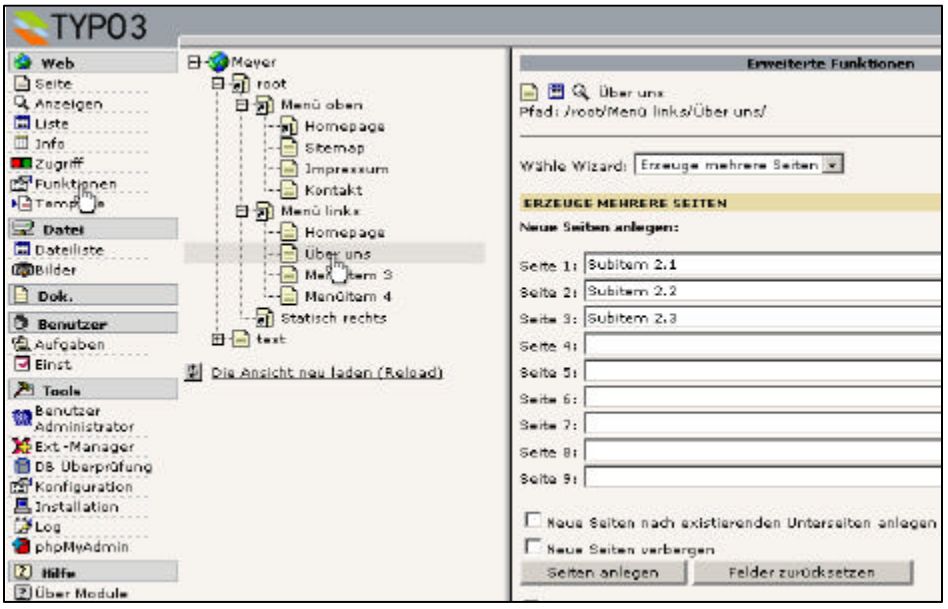
- In Zeile 70 haben wir zur Vereinfachung ausgeklammert. Die Klammerung endet in Zeile 76.
- In Zeile 71 wurde die bisherige Hintergrundfarbe "zwischengespeichert". Die Variable "tmp" ist frei gewählt – auch hätte hier als Variable "meyer" verwendet werden können. Die ursprüngliche Hintergrundfarbe wird später für die Textfarbe benötigt.
- In Zeile 72 wird die Hintergrundfarbe mit einem helleren Grünton überschrieben
- In Zeile 73 wird die gesamte Textebene 10 in die Textebene 5 kopiert. Die neue Textebene 5 liegt unterhalb der Ebene 10. Die Kopie findet hier Anwendung, da fast alle Eigenschaften des Textes übernommen werden sollen – lediglich der Offset als auch die Textfarbe werden sich ändern.
- In Zeile 74 wird die zwischengespeicherte Farbe in die Eigenschaft "fontColor" kopiert. Somit hat fontColor jetzt die gleiche Farbe, wie die ursprüngliche Hintergrundfarbe.
- In Zeile 75 findet die Verschiebung statt – einen Pixel weiter rechts und einen Pixel weiter unten als der Text, der auf der Ebene 10 dargestellt wird.

4.3.10 Eine zweite Menüebene und weitere Zustände hinzufügen

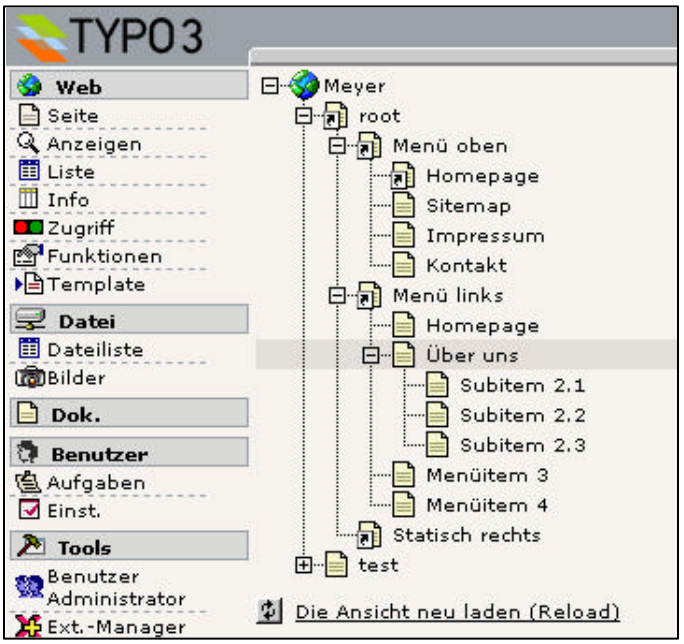
Wir können selbstverständlich auch weitere Menüebenen zu unserem Menü auf der linken Seite hinzufügen. Der Grafiker hat hierzu keine grafischen Vorgaben gemacht. Wir machen uns die Regeln somit selber: Die Menüpunkte sollen eine hellgrüne Hintergrundfarbe haben (wie in der rechten Spalte), der Text soll dunkelgrün erscheinen und jeder Menüeintrag soll durch eine dunkelgrüne Linie voneinander abgetrennt werden.

Unterseiten anlegen

Zunächst müssen wir jedoch noch eine weitere Menüebene hinzufügen. Hierzu legen wir auf der Seite "Über uns" drei Unterseiten an "Subtitem 2.1", "Subitem 2.2" sowie "Subitem 2.3":



Im Seitenbaum sollten wir nun unterhalb von der Seite "Über uns" drei neue Seiten sehen können:



Template erweitern

Versuchen wir nun unsere zweite Menüebene zunächst optisch identisch mit der ersten Menüebene umzusetzen:

```
44        MENU_LINKS = HMENU
          [...]
47        MENU_LINKS.1 = GMENU
48        MENU_LINKS.1.NO = 1
49        MENU_LINKS.1.NO {
             [...]
53           10 = TEXT
             [...]
61           20 = IMAGE
             [...]
68        }
69        MENU_LINKS.1.RO < .MENU_LINKS.1.NO
             [...]
76        }
77        MENU_LINKS.2 < .MENU_LINKS.1
```

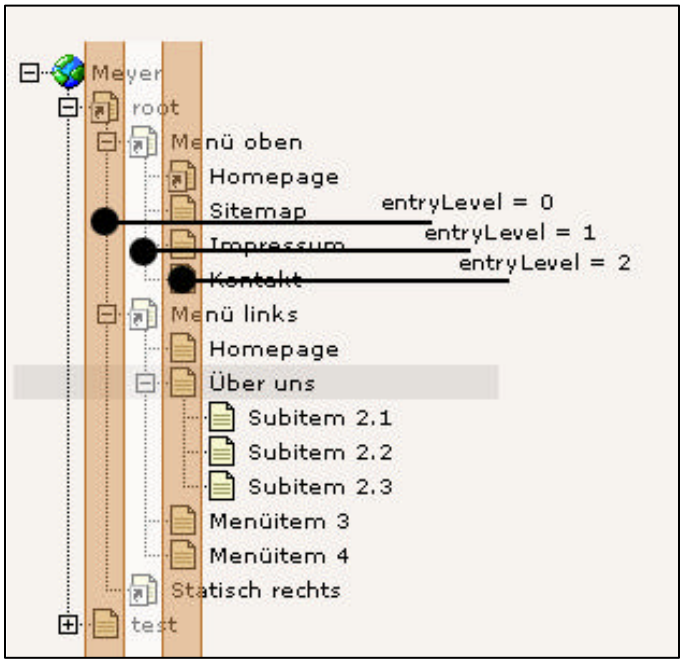
- In Zeile 77 wird auf die zweite Ebene die Beschreibung der ersten Ebene kopiert. Die Nummer "1, 2,..." zum Beispiel bei "MENU_LINKS.2", geben hierbei die Menüebene an. Es wird lediglich die optische Beschreibung kopiert – nicht der Inhalt der Menüeinträge. Somit hat die zweite Menüebene die identische Beschreibung wie die erste Menüebene.

Ein Blick in das Frontend wird jedoch keine zweite Menüebene für die Seite "Über uns" erscheinen lassen, obwohl in unserem Template soweit alles in Ordnung ist.

entryLevel für weitere Menüebenen setzen

Um bei der Verwendung der special-Eigenschaft des HMENU-Objektes weitere Ebenen darstellen zu können, ist es erforderlich, dass wir eine Eigenschaft "entryLevel" setzen. entryLevel gibt an, auf welcher Ebene sich der Ausgangspunkt, in unserem Fall also die Seite "Menü links", befindet und ist ebenfalls eine Eigenschaft des HMENU-Objektes.

Unsere Seite "root" befindet sich auf der Ebene 0, die Seiten "Menü links", "Menü oben" auf der Ebene 1. Folgende Grafik soll dieses veranschaulichen:

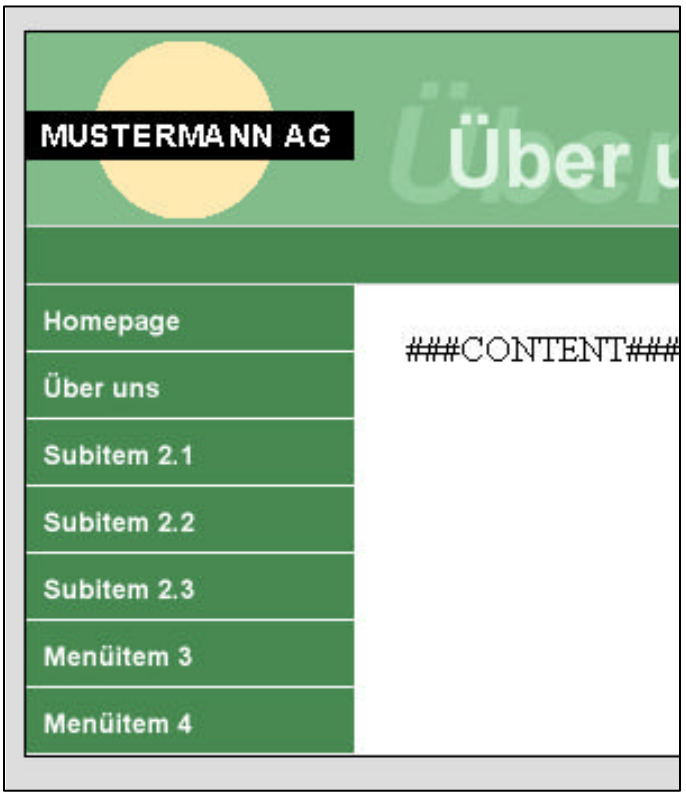


In Typo3script haben wir durch "special.value = 9" angegeben, dass das Menü von der Seite 9 ("Menü link") aufgebaut werden soll. Die Seite Menü links befindet sich in der Baumdarstellung auf der Ebene 1.

Wir erweitern nun unser Template um die Eigenschaft entryLevel:

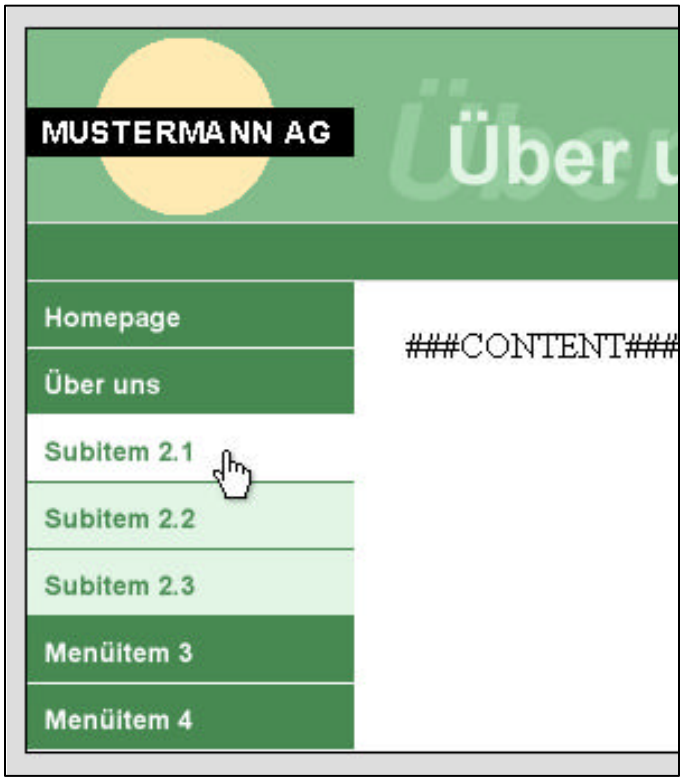
```
43        MENU_LINKS = HMENU
44        MENU_LINKS.special = directory
45        MENU_LINKS.special.value = 9
46        MENU_LINKS.entryLevel = 1
```

Ein erneutes Betrachten unserer Seite im Frontend sollte uns nun zufrieden stimmen:



Für die zweite Menüebene sollten nun aber einige Eigenschafter anders sein: hellgrüner Hintergrund, dunkergrüner Text, dunkelgrüne Trennlinie:

```
44        MENU_LINKS = HMENU
45        MENU_LINKS.special = directory
46        MENU_LINKS.special.value = 9
47        MENU_LINKS.entryLevel = 1
48        MENU_LINKS.1 = GMENU
49        MENU_LINKS.1.NO = 1
50        MENU_LINKS.1.NO {
             [...]
54           10 = TEXT
             [...]
62           20 = IMAGE
             [...]
69        }
70        MENU_LINKS.1.RO < .MENU_LINKS.1.NO
             [...]
76        }
77        MENU_LINKS.2 < .MENU_LINKS.1
78        MENU_LINKS.2 {
79           NO.backColor = #E2F5E5
80           NO.10.fontColor = #478951
81           NO.20.file.backColor = #478951
82           RO < .NO
83           RO.backColor = #FFFFFF
84           RO.5 >
86        }
```



- In Zeile 77 werden die Eigenschaften der ersten Ebene in die zweite Ebene kopiert. Die zweite Ebene ist damit, sofern der entryLevel richtig gesetzt ist, funktionsfähig.

- In Zeile 79 wird die Eigenschaft "backColor" für den normalen Zustand überschrieben und auf ein helles Grün gesetzt.

- In Zeile 80 wird für die Textebene 10 die Schriftfarbe auf ein dunkles Grün gesetzt.

- In Zeile 81 wird die Farbe für die Linie auf ein dunkles Grün gesetzt.

- In Zeile 82 werden die Eigenschaften des normalen Zustands in den RollOver-Zustand kopiert. Da der RollOver-Zustand jedoch schon existiert (wurde in Zeile 77 mitkopiert), findet hier nur eine Überschreibung der Eigenschaften statt.

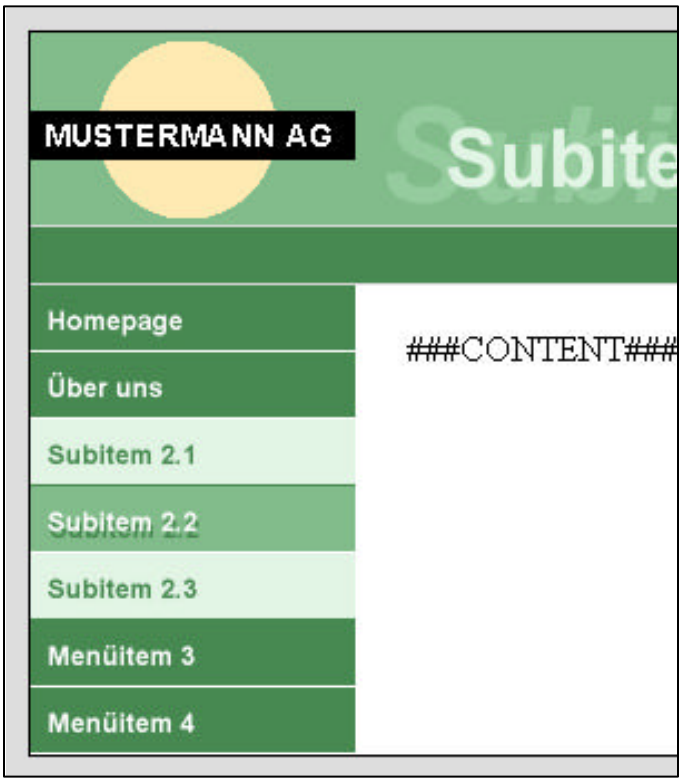
- In Zeile 83 wird die Hinetrgrundfarbe auf weiß gesetzt.

- In Zeile 84 wird die Ebene 5 des RO-Zustands wieder gelöscht. Woher kommt diese Ebene 5? Bei der ersten Menü-Ebene haben wir für den Zustand RollOver angegeben, dass die Ebene 5 den "versetzten Schatten" in dunkelgrünem Text sein soll. In Zeile 77 haben wir auch den Zustand RO kopiert, in Zeile 82 haben wir die bestehenden Eigenschaften überschrieben – die Ebene 5 wurde jedoch nicht entfernt.

Nachträglich den Zustand CUR hinzufügen

Fügen wir nun noch nachträglich den Zustand "CUR" (=Current, die aktuelle Seite) mit in unsere Präsentation ein: CUR soll ein identisches Aussehen haben, wie der jeweilige Zustand RO:

```
44        MENU_LINKS = HMENU
45        MENU_LINKS.special = directory
46        MENU_LINKS.special.value = 9
47        MENU_LINKS.entryLevel = 1
48        MENU_LINKS.1 = GMENU
49        MENU_LINKS.1.NO = 1
50        MENU_LINKS.1.NO {
             [...]
69        }
70        MENU_LINKS.1.RO < .MENU_LINKS.1.NO
             [...]
77        MENU_LINKS.1.CUR < .MENU_LINKS.1.RO
78        MENU_LINKS.2 < .MENU_LINKS.1
```



Wie sehen, dass für den Menüpunkt "Über uns" der Zustand CUR wie gewünscht dargestellt wird (Sofern "Über uns" die aktuelle Seite ist) – für die Untermenüpunkte in der zweiten Ebene, wie oben grafisch dargestellt, wurde jedoch nicht der RO-Zustand der zweiten Ebene verwendet (weißer Hintergrund), sondern der ersten Ebene.

Wie kommt es hierzu? Beim Kopieren der ersten Ebene in die zweite Ebene (Zeile 78) wird ebenfalls der Zustand CUR mitkopiert und steht mit seinen Eigenschaften nun auch in der zweiten Ebene zur Verfügung. Wir haben in der zweiten Ebene nun zwar die Eigenschaften für die Zustände NO und RO angepasst, jedoch noch nicht für den Zustand CUR:

```
77        MENU_LINKS.1.CUR < .MENU_LINKS.1.RO
78        MENU_LINKS.2 < .MENU_LINKS.1
79        MENU_LINKS.2 {
80           NO.backColor = #E2F5E5
81           NO.10.fontColor = #478951
82           NO.20.file.backColor = #478951
83           RO < .NO
84           RO.backColor = #FFFFFF
85           RO.5 >
86           CUR < .RO
87        }
```

Letzter Feinschliff: Fehlender Zeilenumbruch

Wir könnten fast zufrieden sein, gäbe es nicht noch eine kleine Unschönheit. Warum wird das Menü überhaupt untereinander dargestellt? Wie würde man denn ein Menü realisieren, dass nebeneinander stehen soll?

Ein Blick in den HTML-Quelltext verrät uns, dass die Grafiken tatsächlich ohne Zeilenumbruch dargestellt werden:

```
[...]
<a href="17.0.html" onfocus="blurLink(this);" onmouseover="over('img17_6842_1');"
        onmouseout="out('img17_6842_1');">
<img src="typo3temp/m_815d6694ce.gif" width="146" height="30" border="0" alt="Über
        uns" name="img17_6842_1" /></a>
<a href="21.0.html" onfocus="blurLink(this);" onmouseover="over('img21_e6b9_0');"
        onmouseout="out('img21_e6b9_0');">
<img src="typo3temp/m_90047474e8.gif"   width="146" height="30" border="0"
        alt="Subitem 2.1" name="img21_e6b9_0" /></a>
[...]
```

Der Zeilenumbruch wird (freundlicherweise) vom Browser eigenmächtig vorgenommen, bedingt durch die Tabellen-Zelle, der wir eine Breite von 146 Pixeln in der Designvorlage zugewiesen haben.

Um den Zeilenenumbruch dennoch "sauber" zu lösen, können wir nach jedem Menüeintrag einen br-Tag einfügen – mittels wrap:

```
44        MENU_LINKS = HMENU
45        MENU_LINKS.special = directory
46        MENU_LINKS.special.value = 9
47        MENU_LINKS.entryLevel = 1
48        MENU_LINKS.1 = GMENU
49        MENU_LINKS.1.NO = 1
50        MENU_LINKS.1.NO {
51           wrap = |<br>
             [...]
70        }
```

Das gesamte Template des linken Menüs:

```
01      seite = PAGE
        [...]
44      MENU_LINKS = HMENU
45      MENU_LINKS.special = directory
46      MENU_LINKS.special.value = 9
47      MENU_LINKS.entryLevel = 1
48      MENU_LINKS.1 = GMENU
49      MENU_LINKS.1.NO = 1
50      MENU_LINKS.1.NO {
51         wrap = |<br>
52         XY = 146, 30
53         backColor = #478951
54
55         10 = TEXT
56         10.text.field = title
57         10.fontColor = #FFFFFF
58         10.fontFile = fileadmin/fonts/arialb.ttf
59         10.fontSize = 12
60         10.niceText = 1
61         10.offset = 7, 19
62
63         20 = IMAGE
64         20.file = GIFBUILDER
65         20.file {
66            XY = 146,1
67            backColor = #FFFFFF
68         }
69         20.offset = 0, 29
70      }
71      MENU_LINKS.1.RO < .MENU_LINKS.1.NO
72      MENU_LINKS.1.RO {
73         tmp < .backColor
74         backColor = #82BC8B
75         5 < .10
76         5.fontColor < .tmp
77         5.offset = 8,22
78      }
79
80      MENU_LINKS.1.CUR < .MENU_LINKS.1.RO
81      MENU_LINKS.2 < .MENU_LINKS.1
82      MENU_LINKS.2 {
83         NO.backColor = #E2F5E5
84         NO.10.fontColor = #478951
85         NO.20.file.backColor = #478951
86         RO < .NO
87         RO.backColor = #FFFFFF
88         RO.5 >
89         CUR < .RO
90      }
```

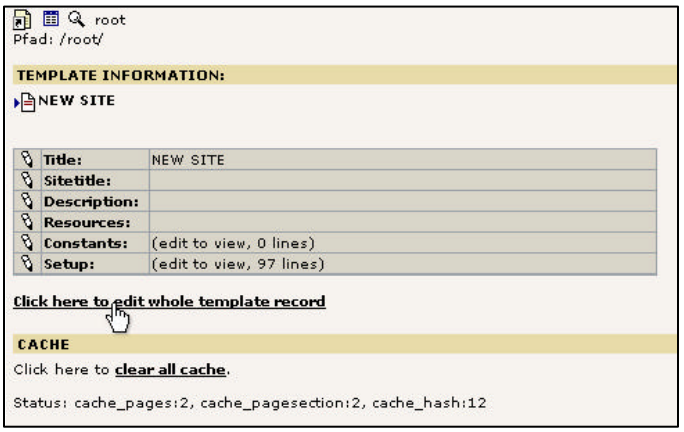
4.3.11 CONTENT: Inhalte ausgeben [mit content(default)]

Nachdem wir unsere Menüs erzeugt haben, können wir uns nun mit der Thematik "Inhalte ausgeben" beschäftigen. Im Kapitel 3.8 wurde bereits das Objekt CONTENT vorgestellt.

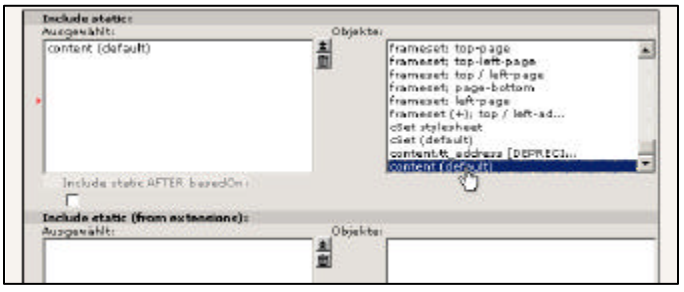
4.3.11.1 Vorbereitung: Statisches Template inkludieren

In Kapitel 3.8 wurde das Prinzip gezeigt, wie Inhalte ausgegeben werden können. Dieser Weg war sehr mühselig und umfangreich. Typo3 hält einen fertigen TypoScript-Code (über 1.000 Zeilen TypoScript) bereit, mit dessen Grundkonfiguration wir arbeiten, und diese anpassen können.

Wir inkludieren nun zu unserem Template ein sogenanntes "statisches Template" mit dem Namen "content (default)". Hierzu klicken wir auf unserem Template ganz unten auf den Textlink "Click here to edit whole template record":

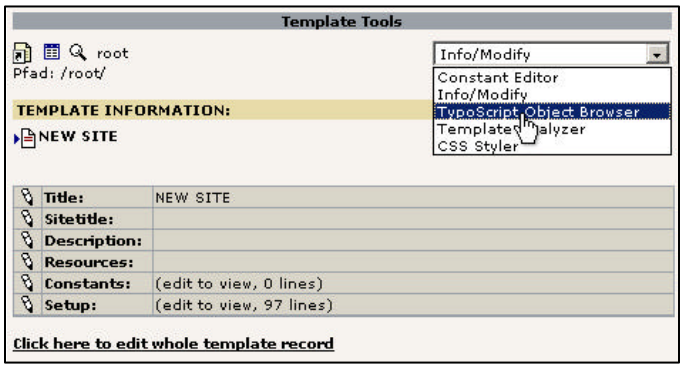


Es öffnet sich eine umfangreiche Template-Maske. Nähere Informationen zu dieser Maske erhalten Sie im Kapitel 2.3. Fügen Sie im Abschnitt "Include Static" auf der rechten Seite den letzten Menüpunkt "content (default)" zu Ihrem Template hinzu und speichern Sie das Template.

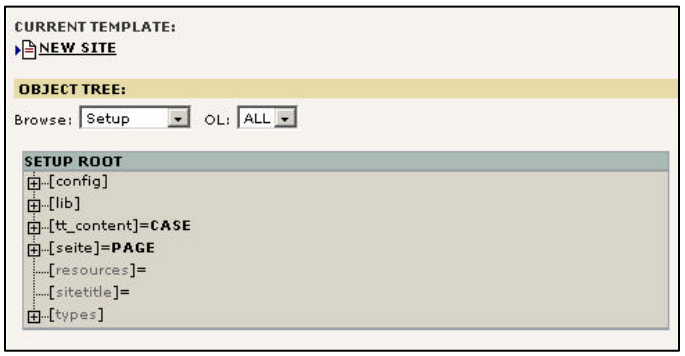


4.3.11.2 Analyse: content (default) unter die Lupe genommen

Wir nehmen nun das inkludierte statische Template unter die Lupe um zu verstehen, wass wir denn überhaupt inkludiert haben. Dieses läßt sich am Besten mit dem Objekt Browser betrachten. Hierzu wählen wir auf unserer Template-Seite rechts oben den Menüpunkt "TypoScript Object Browser" aus:



Der rechte Bereich wird neu geladen und es öffnet sich eine Baumdarstellung unseres Templates. Nähere Informationen im Kapitel 2.4.



Gehen Sie, um zu sehen, was der Object Browser alles enthält, in den Abschnitt "seite" sowie die Unterseiten. Sie werden feststellen, dass hier Ihr TypoScript-Code vollständig abgebildet wird. Sie können auch im Object Browser editieren, indem Sie direkt auf eine Eigenschaft klicken. Das Resultat wird in Ihrem Template gespeichert, sichtbar z.B. im Template-Feld "Setup".

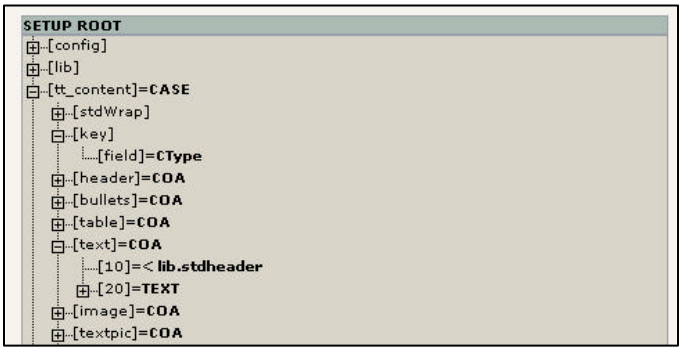
Bei Typo3-Versionen vor 3.6.0 müssen Sie, damit Sie im Objekt Browser editieren können, weiter unten die Option "Enabled object links" aktivieren.

4.3.11.2.1 tt_content

Wir möchten uns hier nun aber nicht mit unserem PAGE-Objekt "seite" beschäftigen, sondern vielmehr mit dem Abschnitt "tt_content". Wie bereits im Kapitel 3.8 erläutert, werden Seiteninhalte in der Tabelle "tt_content" gespeichert. In TypoScript findet diese Tabellenbezeichnung wieder Anwendung.

Wie Sie sehen können, ist tt_content eine Instanz des CASE-Objektes (Kapitel 3.5). Die Case-Abfrage wird auf dem Datenbankfeld "CType" ausgeführt. CType steht für "Content-Typ": Sie haben beim Anlegen eines Seiteninhaltes die Möglichkeit, den Typ des Seiteninhaltes auszuwählen, wie z.B. Text, Text mit Bild etc. Hinter jeder dieser Content-Typen steht ein Wert, der in der Tabelle "tt_content" im Feld "CType" abgespeichert wird: Für den Content-Typ "Text" wird zum Beispiel "text" gespeichert, für "Text mit Bild" "textpic" usw.

Anhand der in CType gespeicherten Werte wird somit eine Case-Abfrage gestartet. Nehmen wir an, wir hätten einen Seiteninhalt vom Typ "Text" erstellt. Die Case-Abfrage würde somit in den Abschnitt "text" springen.



"tt_content.text" ist wiederum eine Instanz des Objektes "COA" (Kapitel 3.4). COA ermöglicht es uns, dass auf einem Objekt mehrere Objekte in sortierter Reihenfolge abgelegt werden können.

Wie können hier sehen, dass an der Position 10 eine absolute Kopie von "lib.stdheader" erfolgt. "lib.stdheader" enthält die Überschrift des Seiteninhaltes. Nähere Informationen hierzu folgen an späterer Stelle.

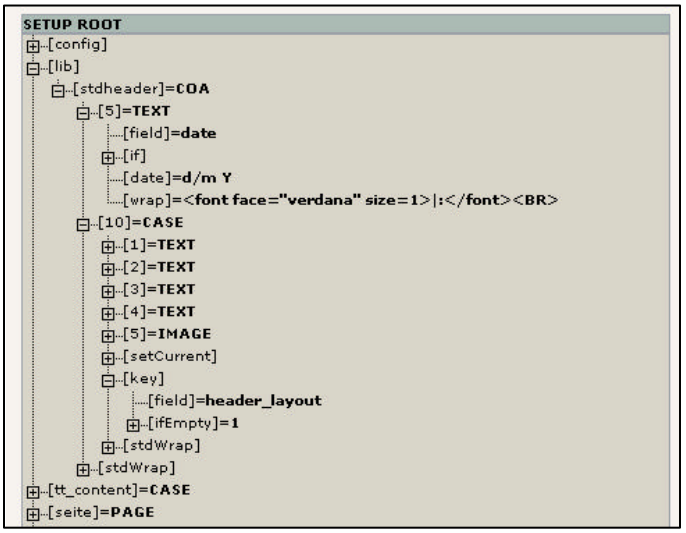
An der Position 20 wird eine Instanz des TEXT-Objektes erzeugt. Durch die Funktion "field = bodytext" wird der in dem Datenbankfeld "bodytext" enthaltene Inhalt ausgegeben. Sonstige weitere Funktionen und Eigenschaften, wie z.B. "textStyle" ermöglichen, zusätzliche Parameter, die im Seiteninhalt angegeben wurden (z.B. Schriftart, Schriftgröße etc.) auszulesen und zu verwenden. Die Funktion "parseFunc" ermöglicht eine Manipulation und Überprüfung des im Datenbankfeld "bodytext" gespeicherten Inhaltes.

4.3.11.2.2 lib.stdheader

Wie bereits oben erwähnt, wurde bei "tt_content.text.10" der Inhalt von "lib.stdheader" kopiert.

```
tt_content.text = COA
tt_content.text.10 < lib.stdheader
tt_content.text.20 = TEXT
```

Doch was enthält lib.stdheader? Betrachten wir uns dieses erneut im Objekt Browser und öffnen gleich einige Instanzen:



Ausgeschrieben in TypoScript können wir folgendes lesen:

```
lib.stdheader = COA
lib.stdheader.5 = TEXT
[...]
lib.stdheader.10 = CASE
lib.stdheader.10.key.field = header_layout
lib.stdheader.10.1 = TEXT
[...]
```

Wir kopieren somit in tt_content.text.10 alles das, was unterhalb von lib.stdheader steht. In tt_content.text.10 steht somit nun:

```
tt_content.text.10 = COA
tt_content.text.10.5 = TEXT
[...]
tt_content.text.10.10 = CASE
tt_content.text.10.10.key.field = header_layout
tt_content.text.10.10.1 = TEXT
[...]
```

lib.stdheader ist vom Ansatz des inkludierten statischen Templates "tt_content (default) dafür geeignet, dass Überschrift und Bodytext bei jedem Seiteninhalt voneinander getrennt sind und nicht zwingend eine Einheit bilden. Bei der Anlage eines Seiteninhaltes kann z.B. angegeben werden, wie das Layout für die Überschrift ist. Dieses Layout findet bei jedem Inhaltstyp Anwendung und betrifft in keiner Weise die Darstellung des Bodytextes. Dieses ist häufig auch so gewünscht – für Änderungen an diesem Ansatz folgen nähere Informationen später in diesem Kapitel.

Sehen wir uns zunächst noch einmal lib.stdheader genauer an:

```
lib.stdheader = COA
lib.stdheader.5 = TEXT
[...]
lib.stdheader.10 = CASE
[...]
```

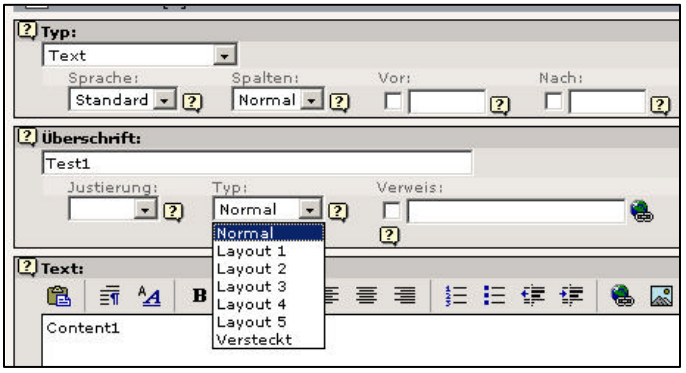
Die Überschrift besteht somit aus (mindestens) 2 Elementen: Dem Element 5 und dem Element 10.

Ein näheres Betrachten von lib.stdheader.5 liefert uns die Information, dass in diesem Bereich offenbar ein Datum angezeigt wird (.field = date), aber nur dann, wenn ein Datenbankfeld "date" auch einen Wert enthält (.if.isTrue.field = date).

In lib.stdheader.10 wird eine CASE-Abfrage auf das Datenbankfeld header_layout vorgenommen (.key.field = header_layout). Die im Objekt Browser sichtbaren Abfragewerte und nun im folgenden auch Instanzen 1-4 sind jeweils Instanzen des TEXT-Objektes, der Abfragewert 5 jedoch ein IMAGE-Objekt. Steht somit zu dem jeweiligen Datensatz im Datenbankfeld "header_layout" eine "1", wird der Abfragewert "1" verwendet und eine Instanz des TEXT-Objektes mit seinen jeweils angegebenen Eigenschaften gebildet. Wird im Datenbankfeld "header_layout" beispielsweise eine "5" gespeichert, so würde eine Instanz des IMAGE-Objektes gebildet werden.

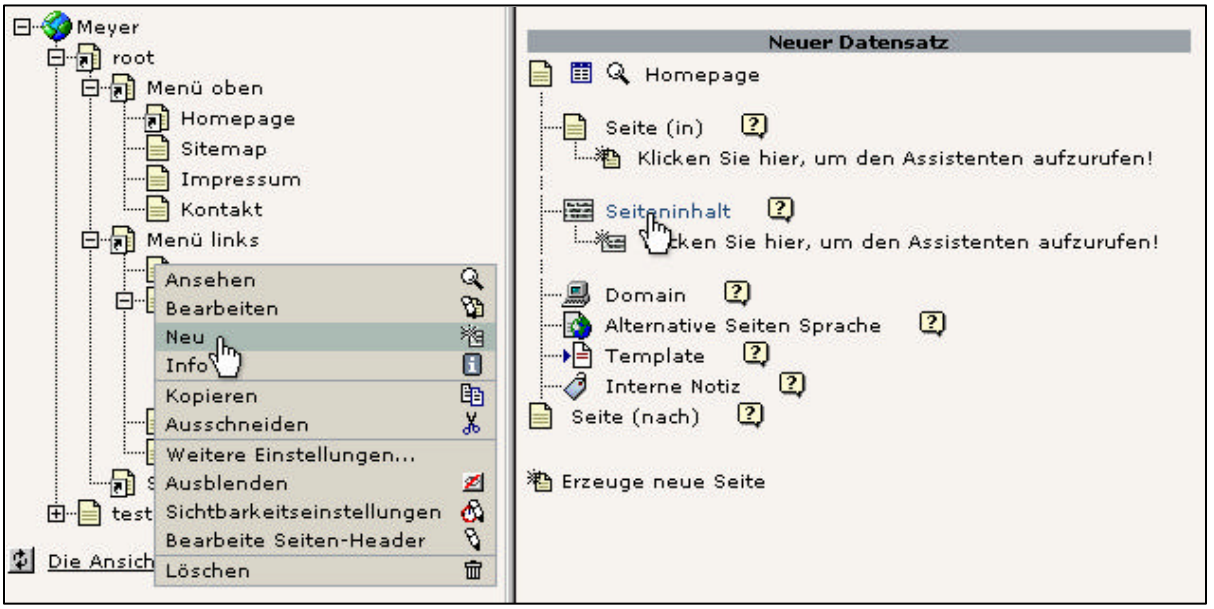
4.3.11.2.3 Das Datenbankfeld "header_layout"

Beim Anlegen bzw. Bearbeiten eines Seiteninhaltes kann das Layout der Überschrift angegeben werden. Die entsprechende Auswahl wird als numerischer Wert in dem Datenbankfeld "header_layout" gespeichert.

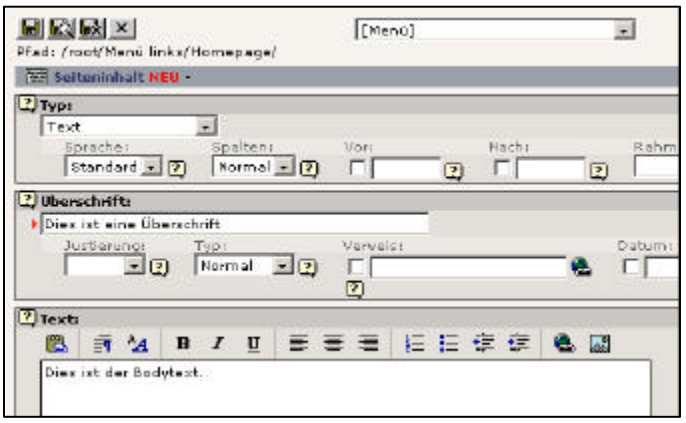


4.3.11.3 Vorbereitung: Einen Seiteninhalt anlegen

Damit wir auf unserer Seite überhaupt Inhalte darstellen können, müssen selbstverständlich auch Inhalte vorhanden sein. Hierzu legen wir z.B. auf unserer Homepage einen Seiteninhalt vom Typ "Text" an. Bitte beachten Sie, dass es mehrere Seiten mit dem Titel "Homepage" gibt, jedoch nur eine davon unsere tatsächliche Homepage ist.



Wir geben nun im Feld Überschrift den Text "Dies ist eine Überschrift" und im Feld Text (RTE) "Dies ist der Bodytext" an. Anschließend speichern wir den Seiteninhalt, indem wir auf das Icon "Speichern und Schließen" klicken.



Weitere Informationen als auch eine andere Herangehensweise zum Anlegen von Seiteninhalten finden Sie im Kapitel 3.8.1.

4.3.11.4 Objekt CONTENT verwenden

Wir erweitern nun unser Template, damit wir den soeben angelegten Inhalt angezeigt bekommen. Der Inhalt soll an dem Marker "CONTENT" dargestellt werden. Beachten Sie bitte, dass sich der folgende Abschnitt innerhalb der geschweiften Klammer von "seite.10.marks" befinden muss:

```
01     seite = PAGE
       [...]

91     CONTENT = CONTENT
92     CONTENT {
93        table = tt_content
94        select.orderBy = sorting
95        select.where = colPos = 0
96     }
```

- In Zeile 91 wird auf dem Marker CONTENT eine Instanz des Content-Objektes gebildet.
- In Zeile 93 wird angegeben, dass der Inhalt aus der Datenbank-Tabelle "tt_content" kommen soll.
- In Zeile 94 und 95 wird die intern erstellte SQL-Anweisung erweitert. Zum Einen geben wir an, dass die Datensätze sortiert werden sollen (Zeile 94), zum Anderen, dass nur Seiteninhalt angezeigt werden sollen, die in der Spalte "Normal" angelegt wurden.

4.3.11.5 Fehleranalyse: Es werden keine Inhalte dargestellt

Falls kein Seiteninhalt dargestellt wird, überprüfen Sie bitte folgende Punkte:

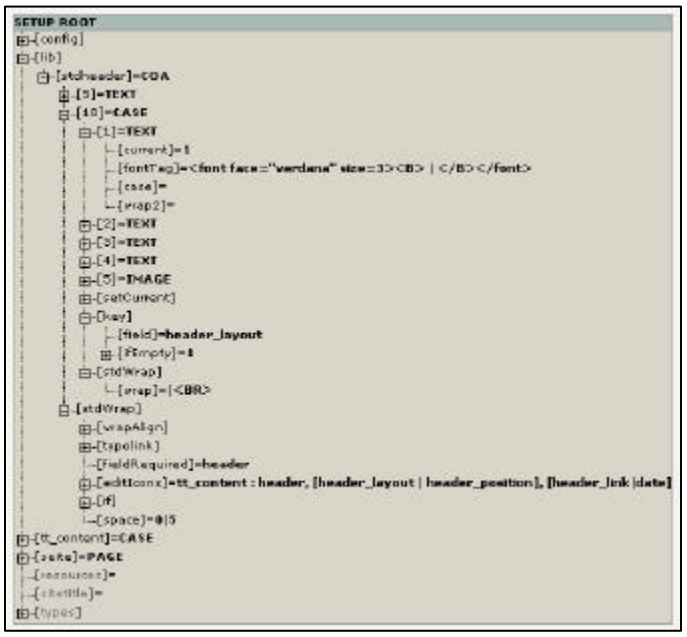
- Wurde für das Template das statische Template "content (default)" inkludiert? (Kapitel 4.3.11.1)
- Überprüfen Sie, ob Sie wirklich die Seite im Frontend betrachten, auf der Sie einen Seiteninhalt angelegt haben.

- Wenn der Marker noch angezeigt wird, haben Sie in TypoScript entweder ein Problem mit der Ausklammerung ("seite.10.marks" wird zum Beispiel nicht vorangestellt) oder aber Sie haben den Markerbezeichner falsch geschrieben (z.B. Groß- und Kleinschreibung).
- Wird der Marker nicht mehr angezeigt, dann sollten Sie die Schreibweise der Objektzuweisung, der Eigenschaften und der Wertzuweisungen überprüfen.

4.3.11.6 Darstellung anpassen, Überschrift

Bei der Darstellung des im Kapitel 4.3.11.4 angezeigten Seiteninhaltes entspricht die Darstellung mit großer Wahrscheinlichkeit nicht unseren Wünschen: Die Überschrift wird in der Schriftart "Verdana" angezeigt, der Bodytext jedoch in der Times (Browser-Standard). Hier sind im Regelfall Anpassungen notwendig.

Betrachten wir uns im Objekt-Browser erneut, wie die Überschrift dargestellt wird:



An der Eigenschaft lib.stdheader.10.1.fontTag können wir sehen, dass hier als Wert "<B> | </B>" zugewiesen wurde. Somit steht für die Überschrift für das Layout "normal" fest, dass diese in der Schrift Verdana und in der Schriftgröße 3 dargestellt werden soll. Wir können dieses einfach im Objekt-Browser an unsere Bedürfnisse anpassen, indem wir auf die Eigenschaft "fontTag" klicken und hier unseren gewünschten Wert angeben.

Die Darstellung selber definieren: TEXT

Wir können aber auch die Darstellung der Überschrift vollständig selber definieren. Hierzu schreiben wir außerhalb der geschweiften Klammern, idealerweise am Ende unseres Templates, folgende eigenständige Definition:

```
100    lib.stdheader >
101    lib.stdheader = CASE
102    lib.stdheader {
103       key.field = header_layout
104       1 = TEXT
105       1.field = header
106       1.wrap = <font face="Arial" size="2"><b> | </b></font><br>
107       default < .1
108    }
```

- In Zeile 100 wird der gesamte Inhalt von lib.stdheader gelöscht. Damit steht lib.stdheader nicht mehr zur Verfügung.
- In Zeile 101 bilden wir an lib.stdheader eine Instanz des CASE-Objektes. Dieses ermöglicht uns zu einem späteren Zeitpunkt nach dem Datenbankfeld "header_layout" (Zeile 103) zu unterscheiden.
- In Zeile 104 geben wir an, dass für den in dem Datenbankfeld "header_layout" gespeicherten Wert "1" ein TEXT-Objekt verwendet werden soll (Kapitel 4.3.11.2.3).
- In den Zeilen 105 und 106 geben wir für das TEXT-Objekt entsprechende Eigenschaften an, so z.B. auch, dass die Überschrift in Arial und in Fett erscheinen soll.
- In Zeile 107 kopieren wir in die default-Eigenschaft des CASE-Objektes unsere TEXT-Instanz aus "1". Ist in dem Datenbankfeld "header_layout" zum Beispiel eine "4" (Layout 4) gespeichert, so würde die default-Instanz hierauf reagieren, da wir für den gespeicherten Wert "4" keine explizite Angabe gemacht haben.

Die Darstellung selber definieren: IMAGE

Zusätzlich zu unserer bestehenden Überschrift für das Layout 1 können wir uns z.B. noch eine Überschrift für das Layout 2 erzeugen, die grafisch erzeugt wird:

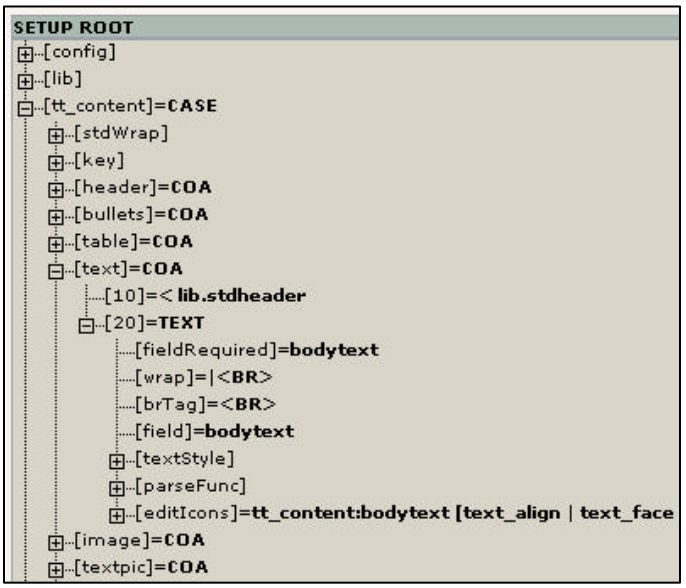
```
100    lib.stdheader >
101    lib.stdheader = CASE
102    lib.stdheader {
103       key.field = header_layout
104       1 = TEXT
105       1.field = header
106       1.wrap = <font face="Arial" size="2"><b> | </b></font><br>
107       2 = IMAGE
108       2.file = GIFBUILDER
109       2.file {
110          XY = [10.w]+10, 28
111          backColor = white
112          10 = TEXT
113          10.text.field = header
114          10.fontColor = #478951
115          10.fontFile = fileadmin/fonts/arialb.ttf
116          10.fontSize = 16
117          10.offset = 1, 22
118          10.niceText = 1
119       }
120       2.wrap = |<br>
121       default < .1
122    }
```

Damit wir nun aber auch die Überschrift im "Layout 2" angezeigt bekommen, müssen wir für den Seiteninhalt das Layout für die Überschrift noch entsprechend ändern (Kapitel 4.3.11.2.3).

4.3.11.7 Darstellung anpassen: Bodytext

Nachdem wir nun unsere Überschrift angepasst haben, werden wir im Folgenden den Bodytext anpassen – zunächst nur für den Seiteninhaltstyp "Text".

Die Definition des inkludierten TypoScripts für den Bodytext können wir uns wieder im Objekt-Browser ansehen. Öffnen wir hierzu im Objekt Browser den Abschnitt "tt_content.text.20":



An der Position tt_content.text.20 wird eine Instanz des TEXT-Objekts erstellt. Die verwendeten Eigenschaften sollen hier nun kurz vorgestellt werden:

- "fieldRequired" gibt an, dass im Datenbankfeld "bodytext" ein Wert enthalten sein muss. Ist kein Wert in Bodytext enthalten, wird dieser Abschnitt nicht ausgeführt. Diese Abfrage ist in Bezug auf den folgenden wrap wichtig.
- "brTag" gibt an, welcher Wert für einen Zeilenumbruch (\n) in der Datenbank verwendet werden soll. Wir also in der Datenbank ein Zeilenumbruch gefunden, wird dieser mit dem angegebenen HTML-Tag, hier also <BR>, ersetzt.
- Mit der Eigenschaft "textStyle" kann dynamisch der Font-Tag erzeugt werden. Diese Eigenschaft ist nur dann hilfreich, wenn Redakteure die Möglichkeit erhalten sollen, selbst festzulegen, ob der aktuelle Text in der Schriftart "Times" oder z.B. "Verdana" angezeigt werden soll. Ansonsten ist es ratsam, auf diese Eigenschaft zu verzichten und die Darstellung direkt im wrap zu definieren.
- Die Funktion "parseFunc" übernimmt die Überprüfung des eingelesenen Datenbankwertes. Hier wird auf verwendete HTML-Tags hin überprüft. Sehr nützliche Funktion.
- Mit "editIcons" können für das adminPanel im Frontend, welches nur für eingeloggte Backend-Benutzer bei entsprechender Konfiguration sichtbar ist, Einstellung vorgenommen werden.

Um nun den Bodytext anzupassen, können wir uns von obiger Darstellung lösen und unsere eigene Darstellung "bauen". Hierzu erweitern wir auf oberster Ebene außerhalb des PAGE-Objekt-Abschnitts (idealerweise am Ende unseres Templates) unser Template:

```
123    tmp_tt_content_text < tt_content.text
124    tt_content.text >
125    tt_content.text = COA
126    tt_content.text.10 < lib.stdheader
127    tt_content.text.20 = TEXT
128    tt_content.text.20 {
129       field = bodytext
130       fieldRequired = bodytext
131       wrap = <font face="Arial" size="2">|</font><br>
132       parseFunc < tmp_tt_content_text.20.parseFunc
133       editIcons < tmp_tt_content_text.20.editIcons
134    }
```

- In Zeile 123 wird das Original von tt_content.text temporär in "tmp_tt_content_text" zwischengespeichert.
- In Zeile 124 wird tt_content.text gelöscht: Die Definition, wie ein Seiteninhalt vom Typo3 "Text" auszusehen hat, existiert somit nicht mehr.
- In Zeile 125 wird für den Seiteninhalt "Text" eine Instanz des Objektes COA erstellt. Dieses ist erforderlich, da Überschrift und Inhalt getrennt sind und sich für die Überschrift ein Layout auswählen läßt.
- In Zeile 126 wird an die Position 10 aus der temporären Kopie, die in Zeile 123 erstellt wurde, nur die Überschrift kopiert. Die Definition der Überschrift haben wir bereits im vorherigen Kapitel 4.3.11.6 vorgenommen.
- In den Zeilen 127 bis 134 wird die Darstellung des Text-Objektes neu definiert. Hierbei werden jedoch die bereits erstellten Definitionen für die Funktionen parseFunc sowie editIcons aus der erstellten Kopie des Originals (Zeile 123) übernommen.

4.3.12 Rechte Spalte: Inhalte darstellen

Der Marker "###RECHTS###" wird ähnlich dargestellt, wie der Marker ###CONTENT###. Einziger Unterschied ist die Angabe der Eigenschaft select.where des CONTENT-Objektes.

```
01     seite = PAGE
       [...]

91     CONTENT = CONTENT
92     CONTENT {
93        table = tt_content
94        select.orderBy = sorting
95        select.where = colPos = 0
96     }
97     RECHTS = CONTENT
98     RECHTS {
99        table = tt_content
100       select.orderBy = sorting
101       select.where = colPos = 2
102    }
```

4.3.12.1 Die Spalten: colPos

In Typo3 kann der Redakteur Inhalte auf Spalten anlegen. Diese sind für den Redakteur mit den Bezeichner "Links, Normal, Rechts, Rand" gekennzeichnet. Intern wird der jeweilige Wert in dem Datenbankfeld "colPos" der Tabelle tt_content numerisch abgelegt. Mögliche Werte sind 1 bis 4:

Spalte	Gespeicherter Wert
Normal	0
Links	1
Rechts	2
Rand	3

Kapitel 5: Module und eigene Erweiterungen

5.0 Einführung

Typo3 ist seit der Version 3.5.x vollständig modular aufgebaut. In vorherigen Versionen kam Typo3 als "ein Ganzes": News- und Shop-System waren bereits integriert – neue Module konnten nur mit sehr gutem Wissen entwickelt und integriert werden. Seit der Version 3.5.x wurde Typo3 um komfortable Modul-Funktionalitäten erweitern, die dem CMS einen großen Schub nach vorne gegeben haben. Eigene Moduentwicklungen lassen sich nun komfortabel durch einen Klick (oder durchaus auch mal mehrere) installieren.

In der sogenannten "Extension Repository" können Module zentral abgelegt, und so der Öffentlichkeit zur eigenen Verwendung zur Verfügung gestellt werden. Dieses Prinziep hat zu einer sehr großen Ansammlung von Modulen geführt, die jedoch auch Gefahren darstellen: Nicht alle Module sind für den produktiven Einsatz bestimmt und befinden sich noch im Entwicklungsstadium.

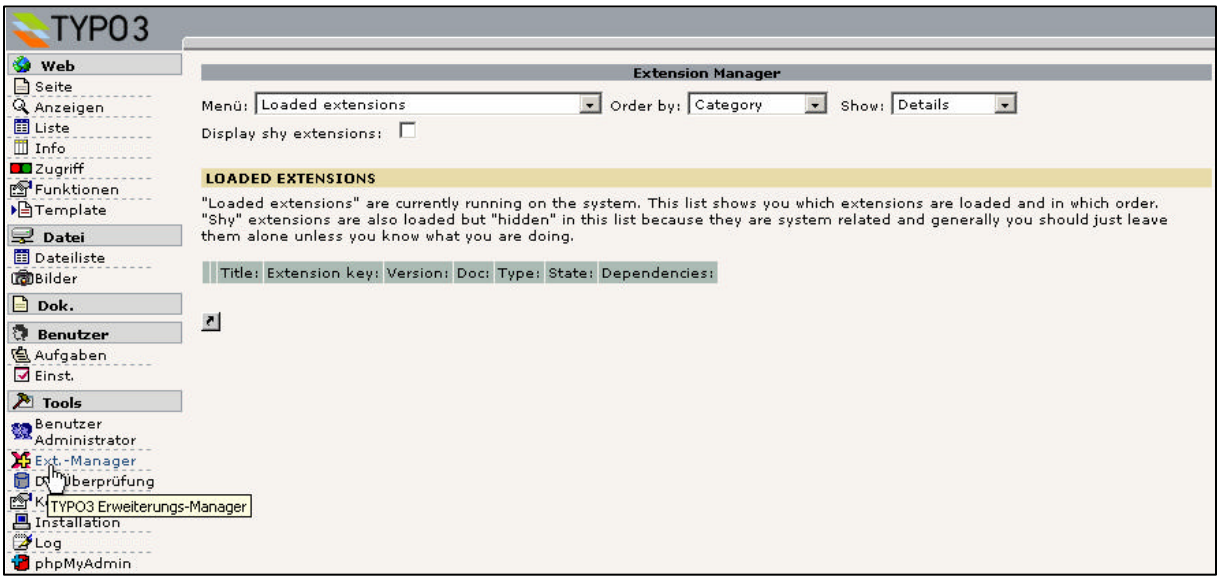
Ebenso kann es seit der Version 3.6.x zu Kompatibilitätsproblemen kommen: Module setzen eine ganz bestimmte Typo3-Version voraus. Steht diese bestimmte Version nicht zur Verfügung, so kann es bei dem einen oder anderen Modul (insbesondere bei Modulen im Beta-Stadium) zu Fehlermeldungen kommen oder die Präsentation läßt sich nicht mehr ohne Fehlermeldungen anzeigen. Im schlimmsten Fall ist ein Login in das Backend aufgrund von Fehlermeldungen nicht mehr möglich. Durch manuelles deinstallieren des Moduls kann der Zugriff auf das Backend wieder ermöglicht werden – jedoch ist dieses immer mit Ärger und Stress verbunden.

Diese Einführung soll nicht davor abschrecken, Module zu verwenden. Jedoch sollten Sie Module nicht in einer Live-Umgebung testen, da dieses unangenehme Folgen haben kann. Testen Sie Module daher möglichst in einer separaten Testumgebung auf Fehler.

5.1 Der Typo3 Erweiterungsmanager

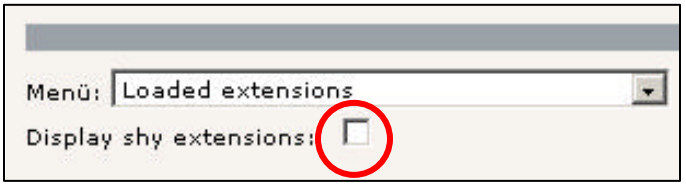
Den Erweiterungs- bzw. Extension-Manager finden wir im linken Menü im Abschnitt "Tools". Ein Klick auf den Menüeintrag "Ext.-Manager" öffnet im rechten Bereich den Typo3-Erweiterungsmanager:

5.1.1 Shy Extensions

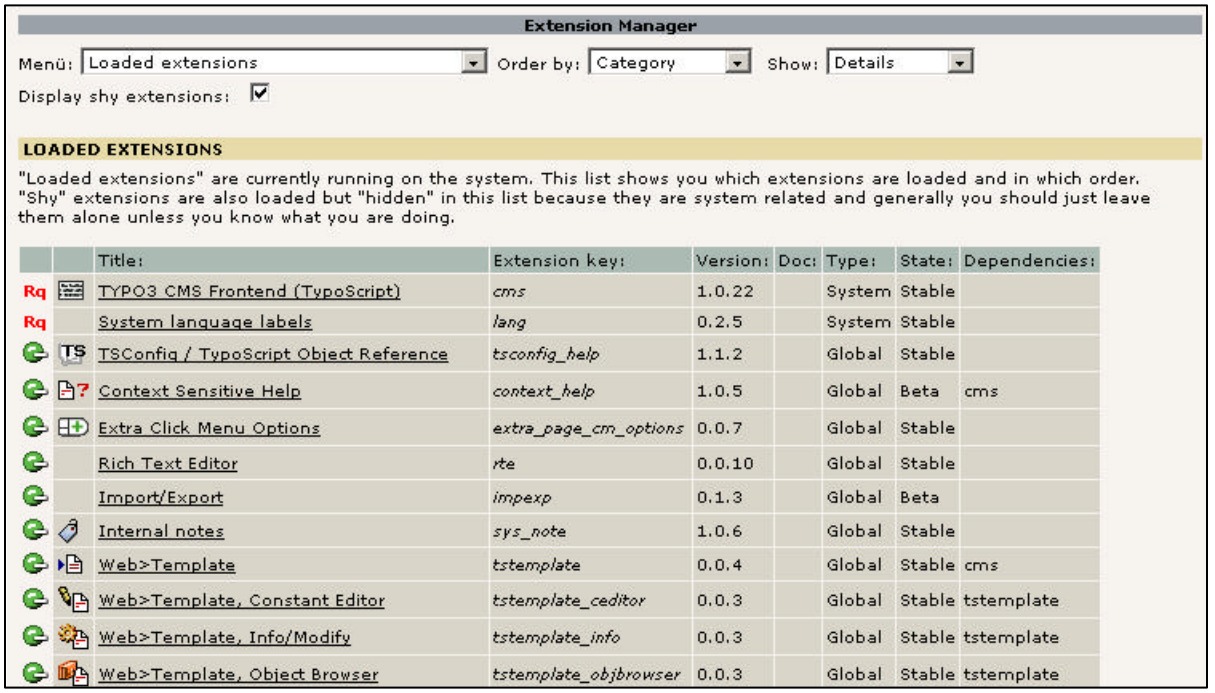


Wir sehen zunächst eine Übersicht übersicht der "geladenen" Module - im obigem Screenshot scheint kein Modul installiert zu sein. Der Schein trügt allerdings. Wie bereits in der Einleitung zu diesem Kapitel (5.0) erwähnt, ist Typo3 selbst fast vollständig modular aufgebaut. Dieser modulare Aufbau ist jedoch nicht dass, was wir als klassische Module erwarten würden.

Für uns sind im Regelfall Module wie z.B. das Shop- oder das News-Modul klassische Erweiterungen. Das jedoch z.B. der RTE (Rich Text Editor) oder aber der Menüpunkt "phpMyAdmin" ebenfalls Module sind, die im "Auslieferungszustand" bereits installiert sind, kann hier nicht unmittelbar erkannt werden. Darum sei hier auf diese Erweiterungen kurz eingegangen, die als "Shy Extensions" bezeichnet werden.



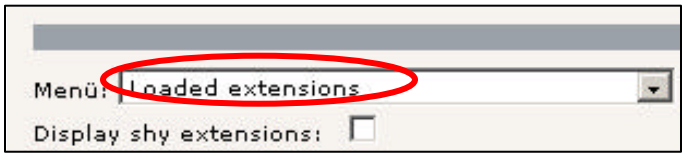
Damit wir diese "versteckten" Extensions angezeigt bekommen, können wir uns diese durch setzen des Häkchens im Feld "Display shy extensions" anzeigen lassen:



Wie wir sehen können, gibt es eine umfangreiche Liste von Modulen, die bereits installiert sind.

5.1.2 Die Auswahlbox "Menü" im Erweiterungsmanager

Links oben im Erweiterungsmanager finden Sie eine Auswahlbox "Menü":



In dieser Auswahlbox finden Sie folgende Menüpunkte:

- Loaded extensions: Unter "Load extensions" finden Sie alle Module, die bereits installiert sind.
- Available extensions to install: Hier finden Sie alle Module, die in Ihrer Typo3-Installation zur Verfügung stehen und mit einem Klick installiert werden können. Wenn eine Modul zur Verfügung steht, dann bedeutet dieses, dass sich die Modul-Dateien entweder in einem Unterordner von /typo3/ext oder von /typo3conf/ext befinden.
- Import extensions from online repository: Steht ein Modul in der Typo3-Installation nicht zur Verfügung, können Sie sich dieses Modul von einem zentralen Server herunterladen. Dieser zentrale Sammelpunkt aller veröffentlichten Module wird als "Extension Repository" bezeichnet.
- Settings: Einstellungen die Sie insbesondere dann benötigen, wenn Sie selbst eigene Module veröffentlichen möchten.

5.1.3 Die Spalten im Erweiterungsmanager

Wir können im Erweiterungsmanager insgesamt 9 Spalten erkennen.

Die erste Spalte beinhaltet entweder ein grünes Symbol mit einem Minuszeichen, ein graues Symbol mit einem Pluszeichen oder aber das Symbol "Rq". Das grüne Symbol mit dem Minuszeichen gibt an, dass das Modul installiert ist und durch einem Klick auf dieses Symbol das Modul deinstalliert werden kann. Das graue Symbol gibt an, dass das Modul in der Typo3-Installation zur Verfügung steht, aber nicht installiert ist. Durch ein Klick auf dieses graue Symbol mit dem Pluszeichen kann das Modul installiert werden. Das Symbol "Rq" (Required) gibt an, dass das Modul nicht deinstalliert werden kann, da es zwingend für den Betrieb von Typo3 benötigt wird. Dies sind im Regelfall nur zwei Module: Das TypoScript-Modul sowie die Sprachdateien.

In der zweiten Spalte wird lediglich ein Symbol für das jeweilige Modul angezeigt. Diese Symbole lassen sich nicht anklicken, auch öffnet sich kein PopUp-Fenster, wie sonst bei Symbolen oder Icons in Typo3 üblich.

Die dritte Spalte beinhaltet die Bezeichnung des Moduls, damit wir wissen, um welches Modul es sich überhaupt handelt, dass wir z.B. gerade deinstallieren möchten.

In der vierten Spalte wird der sogenannte "Extension Key" angezeigt. Der Extension Key ist der eindeutige Bezeichner des Moduls und darf nicht noch einmal verwendert werden. Für Modulentwickler, die Ihre selbst entwickelten Module veröffentlichen möchten, steht unter typo3.org ein Formular zur Verfügung, mit dem man sich einen Extension Key reservieren kann, damit es in der internationalen Entwicklergemeinde zu keinen Konflikten kommt.

In der fünften Spalte wird die Versionsnummer des Moduls angezeigt. Bzgl. der Versionen kann es zu Problemen kommen – eine Version 1.2.3.4 eines Moduls kann z.B. mit einer Typo3-Installation Version 3.5.0 unkompatibel sein. Informationen zu unkompatiblen Modulen finden Sie derzeit nur in den Typo3-Foren.

Die sechste Spalte zeigt an, ob zu diesem Modul eine Dokumentation zur Verfügung steht.

In der siebten Spalte "Type" können Sie erkennen, wo das Modul installiert wurde. Dies ist insbesondere für spätere Updates von der Typo3-Installation interessant. Es genau drei Möglichkeiten von Einträgen: System, Global und Lokal. Bei System-Modulen ist es unerheblich, wo diese genau liegen, da bei einem Update diese Module zwingend ausgetauscht werden müssen. Bei Modulen, die "Global" installiert sind, kann es bei späteren Updates Probleme geben, sofern Änderungen an dem Modul vorgenommen worden sind. Module, die "Lokal" installiert sind, sind updateresistent.
Nähere Informationen zur Updatefähigkeit und zum Unterschied "Lokal / Global" erhalten Sie im Kapitel 5.2

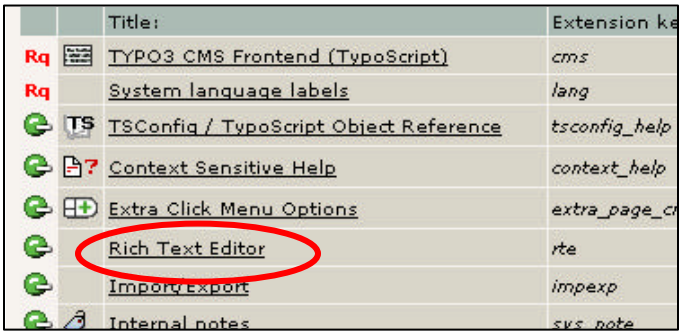
In der achten Spalte wird der Status des Moduls angegeben. Mögliche Werte sind Test, Experimental, Alpha, Beta und Stable. Der Status wird jedoch nicht von einem Gremium

vergeben, sondern direkt vom Entwickler des Moduls selbst. Module mit dem Status "Stable" können also durchaus noch Fehler enthalten, da Sie z.B. nur für eine bestimmte Typo3-Umgebung getestet wurden.

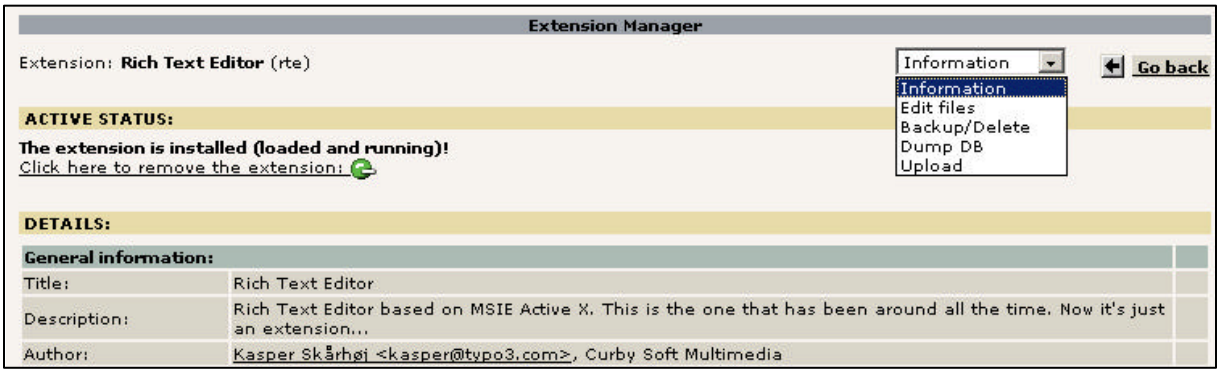
In der neunten Spalte können Sie Abhängigkeiten von anderen Modulen erkennen. Bei den benötigten Modulen wird hierbei der Extension Key angegeben. So läßt sich der Constant Editor nur dann installieren, wenn auch das Modul "tstemplate" installiert ist.

5.1.4 Detailinfos zu den Modulen

Durch einen Klick auf eines der Modultitel erhalten wir nähere Informationen zu dem Menü:



Es werden nun nähere Informationen zum Modul angezeigt, wie z.B. eine Beschreibung, was das Modul überhaupt macht, der Autor etc.

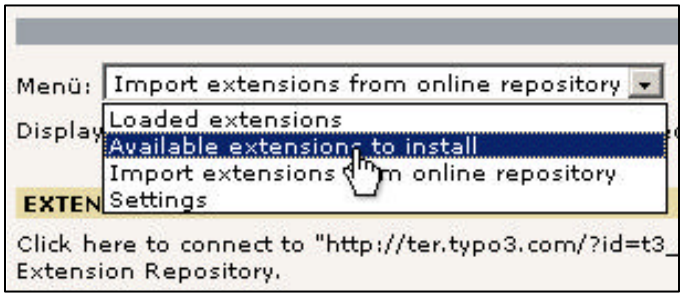


Aus der Auswahlbox rechts oben können noch weitere Bereich ausgewählt werden, wie z.B. der Menüpunkt "Edit Files".

Mit "Edit Files" können Sie nur dann Dateien editieren, wenn dieses in Ihrer Typo3-Installation möglich ist. Mit dem Install-Tool können Sie diese Möglichkeit geben oder beschränken.

5.1.5 Verfügbare Module installieren

Verfügbare Module, die in unserer Typo3-Installation zur Verfügung stehen, können Sie mit einem Klick (bzw. zwei Klicks) installieren. Um eine Übersicht zu erhalten, welche Module überhaupt vorhanden sind, wählen Sie aus der Auswahlbox "Available extensions to install" aus:

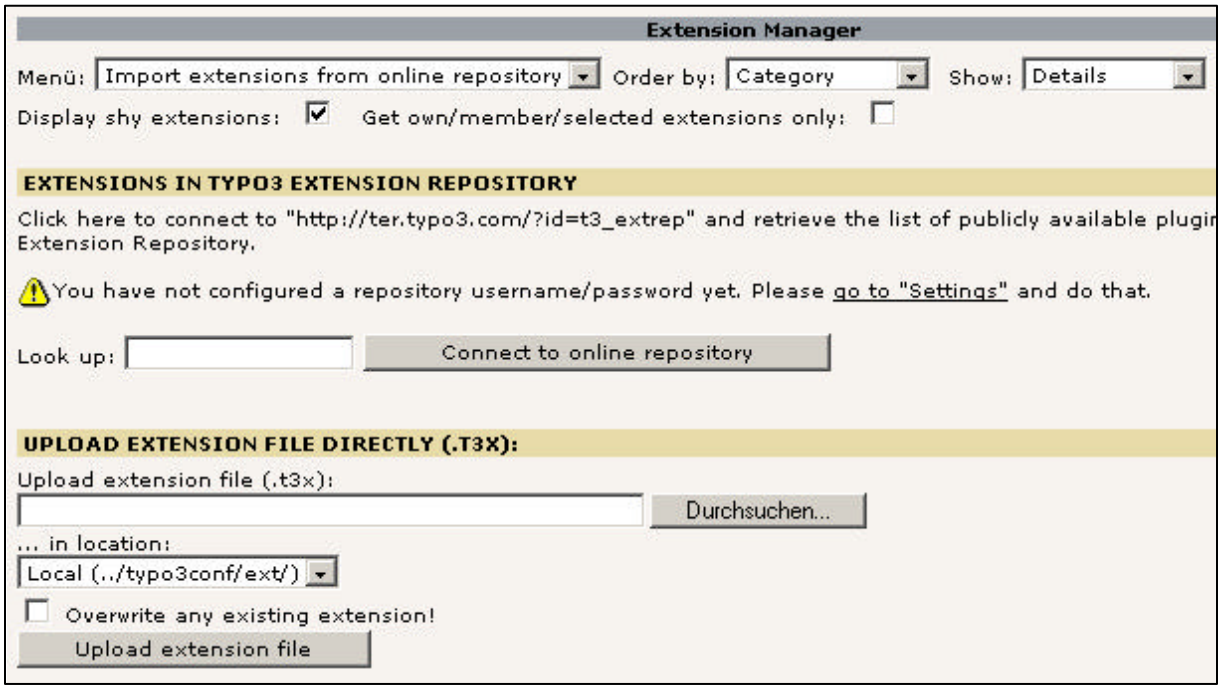


Sie erhalten eine Übersicht der Module, die sich in Ihrer Typo3-Installation befinden. Um ein Modul zu installieren, klicken Sie auf das graue Symbol mit dem Plus-Zeichen. Bei einigen Modulen werden Sie auf einer Folgeseite gefragt, ob Datenbank-Tabellen angelegt werden sollen und der Cache geleert werden darf. Sie können getrost einfach mit "Make Updates" bestätigen, um das Modul entgültig zu installieren.

5.1.6 Module von der Extension Repository herunterladen

Steht ein Modul nicht zur Verfügung, so können Sie dieses von der Extension Repository herunterladen. Die Extension Repository ist ein zentraler Server als Sammelpunkt aller veröffentlichten Module.

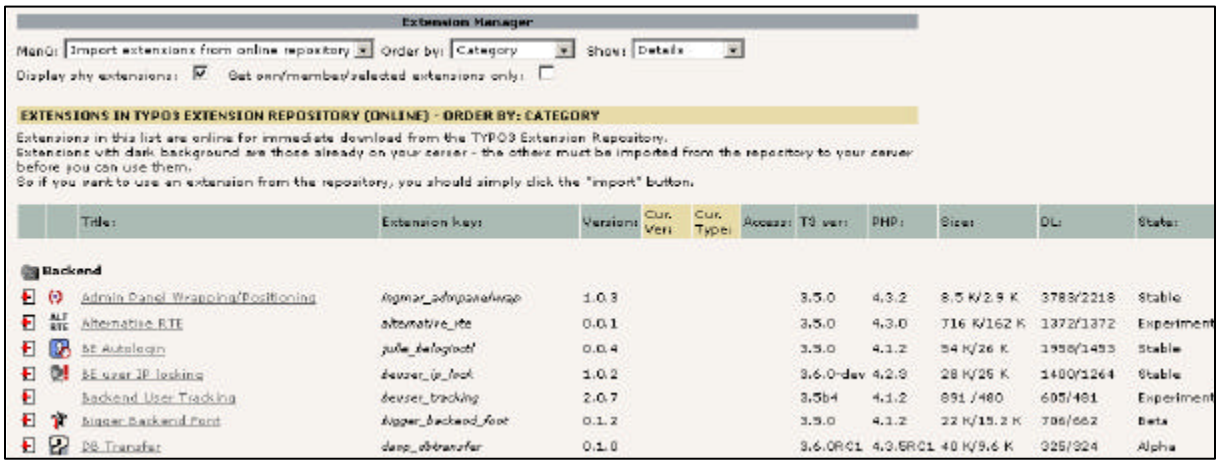
Um eine Verbindung zu diesem zentralen Server aufzubauen, wählen Sie aus der Auswahlbox den Eintrag "Import extensions from online repository":



Sie sehen einen Warnhinweis "You have not configured a repository username/password yet". Dieser Warnhinweis kann mit ruhigem Gewissen vernachlässigt werden und ist insbesondere nur dann interessant, wenn Sie selbst Module veröffentlichen möchten.

Um nun eine Verbindung zur Repository aufzubauen, klicken Sie einfach auf den Button "Connect to online repository". Es wird eine Liste der veröffentlichten Module geladen. Da diese Liste sehr lang ist und täglich neue Module veröffentlicht werden, kann es, je nach Anbindung, einige Sekunden dauern, bis diese Liste vollständig geladen wurde.

Da es sich bei der online Repository um einen zentralen Server handelt, kann es durchaus vorkommen, dass dieser Server nicht zur Verfügung steht. Insbesondere nach Veröffentlichungen in der Presse oder nach dem Erscheinen einer neuen Typo3-Version ist dieser zentrale Server sehr stark ausgelastet. Versuchen Sie es daher bei einem fehlerhaftem Verbindungsversuch einfach zu einem späteren Zeitpunkt noch einmal.



Das Symbol auf der linken Seite gibt an, welche Aktion Sie ausführen können. Wenn Sie mit der Maus über das Symbol fahren, erhalten Sie die entsprechenden Informationen darüber, welche Aktion ausgeführt wird. Mit dem "Download" Symbol kann das Modul z.B. direkt in die eigene Typo3-Installation heruntergeladen werden. Bei dem Mouse-Over erhalten Sie ebenfalls die Information, "wohin" das Modul installiert wird: In das lokale Verzeichnis "typo3conf/ext" (updatefähig) oder in das globale Verzeichnis "typo3/ext".

Achten Sie bitte vor dem Herunterladen darauf, dass das Modul auch für die jeweilige Typo3-Installation geeignet ist. Ein Modul, dass für die Typo3-Version 3.6.1 entwickelt wurde, wird in der Version 3.6.2 vermutlich auch lauffähig sein. Bei Versionsunterschieden kann es durchaus auf einen Versuch ankommen. Gehen Sie aber bitte davon aus, dass das Modul ggf. Ihre Typo3-Umgebung lahmlegt und somit Ihre Präsentation für den Zeitraum, bis das Modul manuell wieder entfernt wurde, unter Umständen nicht erreichbar ist.

5.2 Generelles zur Updatefähigkeit

Sie möchten Ihre Module updatefähig halten? Dann haben Sie schon gedanklich eine gute Voraussetzung dafür schaffen, dass es später, sofern Sie denn Updates tätigen möchten, nicht vor unschönen Problemen stehen, die oftmals erst nach einem getätigtem Update auftreten.

Wie bereits in Kapitel 5.1 erwähnt, gibt es zwei Orte, an denen Typo3-Moduldateien liegen können: Lokal und Global. Der lokale Modul-Ordner liegt in /typo3conf/ext, der globale Ordner in /typo3/ext. Das Modul sollte nicht an beiden Stellen gleichzeigt liegen. Module, die im Ordner /typo3conf/ext liegen, sind updateresistent.

Folgender Hintergrund: Bei einer "leeren" Typo3-Installation werden bereits viele Typo3-Module mitgeliefert und sind zum Teil sogar schon installiert (z.B. phpMyAdmin). Dies deutet darauf hin, dass diese Module Bestandteil des Original-Quelltextes sind, der bei einem Update entweder überschrieben oder ausgetauscht wird. Nehmen Sie also Veränderungen an diesem mitgeliefertem Quelltext vor, so wird dieser Quelltext bei einem Update von Typo3 einfach überschrieben bzw. ausgetauscht.

Mitgelieferte Typo3-Module befinden sich im Ordner /typo3/ext. Dieser Ordner würde, unter Anderem, bei einem Update ausgetauscht werden. Wenn ein Modul updatefähig gehalten werden soll, dann sollte sich das Modul im "lokalen Ordner" /typo3conf/ext befinden.

Doch wie kann man z.B. das Newsmodul vom globalen Ordner in den lokalen Ordner "transferieren"? Hierzu gibt es mehrere Möglichkeiten:

1.) Wenn Sie auf Ihre Typo3-Installation mittels SSH zugreifen können, verschieben Sie einfach den Modulordner /typo3/ext/tt_news in den Ordner /typo3conf/ext/tt_news
2.) Wenn Sie per FTP arbeiten, können Sie das Modul zunächst herunterladen, dann vom Server löschen und später wieder in den Ordner /typo3conf/ext/tt:_news hochladen.
3.) Sie können aber auch über das Backend über den Menüpunkt "Backup/Delete" (siehe Kapitel 5.1.4) das Modul vom Server löschen und neu von der Extension Repository herunterladen.

Änderungen am Quelltext nehmen Sie insbesondere nur dann vor, wenn Sie das Modul um Funktionalitäten erweitern möchten. Anpassungen an dem Design, also Änderungen an einer Designvorlage, können Sie aber auch ohne einem Transferieren aller Moduldateien in den lokalen Ordner updatefähig halten. Dies wird zu 95% der Fall sein und wird vollständig über TypoScript gesteuert.

5.3 Bestehende Frontend-Module integrieren und anpassen

5.3.1 Das News-Modul

5.3.1.1 Das klassische Newsmodul installieren

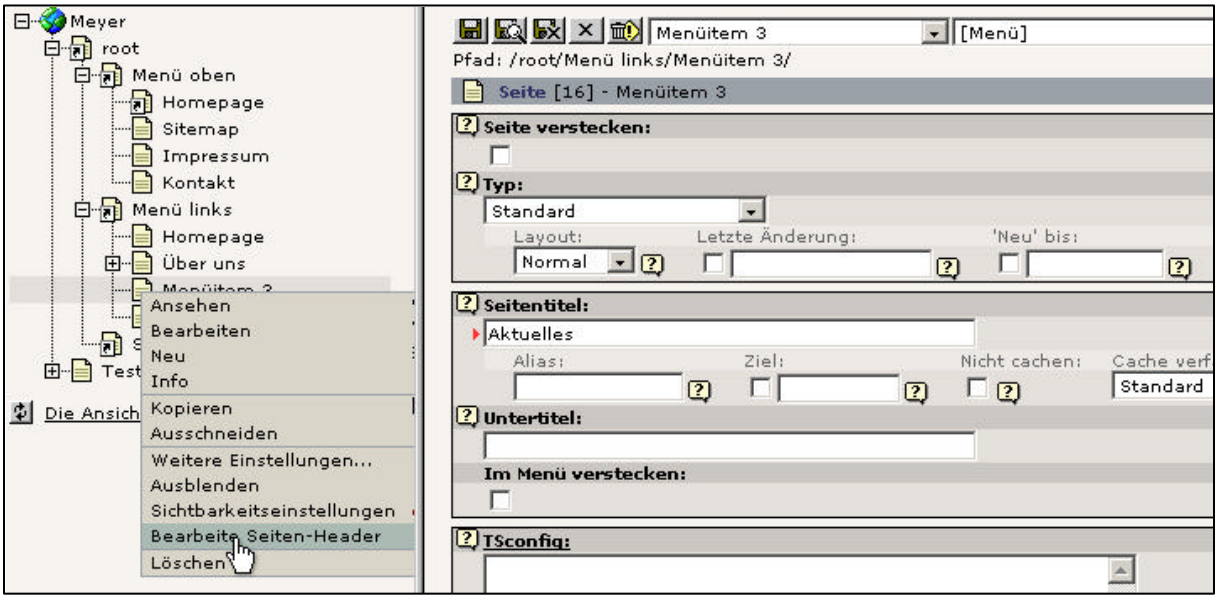
Um das Newsmodul zu installieren, wählen wir im Erweiterungsmanager aus der Auswahlbox den Eintrag "Available Extensions to install" aus. Es öffnet sich eine Liste aller in unserer Typo3-Umgebung bereits vorhandenen Module. Um nun das News-Modul zu installieren, suchen wir im Abschnitt "Frontend Plugins" nach diesem Modul:

Die Version des News-Moduls trägt die Versionsnummer 1.0.4. Eine neue Version des News-Moduls mit der Versionsnummer 1.4 befindet sich derzeit in der Entwicklung. Einige interessante Features sind hinzugekommen. Da diese neue Version 1.4 (noch) nicht Bestandteil der ausgelieferten Version ist, wird hier im Abschnitt 5.3.1 das Newsmodul in der Version 1.0.4 vorgestellt. Diese Vorstellung ist exemplarisch und gilt prinzipiell für alle Frontend-Module.

Um nun das News-Modul zu installieren, klicken wir das graue Plus-Symbol. Es folgt eine Seite, in der wir gefragt werden, ob zusätzliche Tabellen angelegt werden sollen und ob der Cache gelöscht werden darf. Dieses können Sie bestätigen, in dem Sie auf den Button "Make Updates" klicken:
Das News-Modul wurde nun installiert – eine direkte Veränderung im Backend läßt sich auf den ersten Blick nicht erkennen.

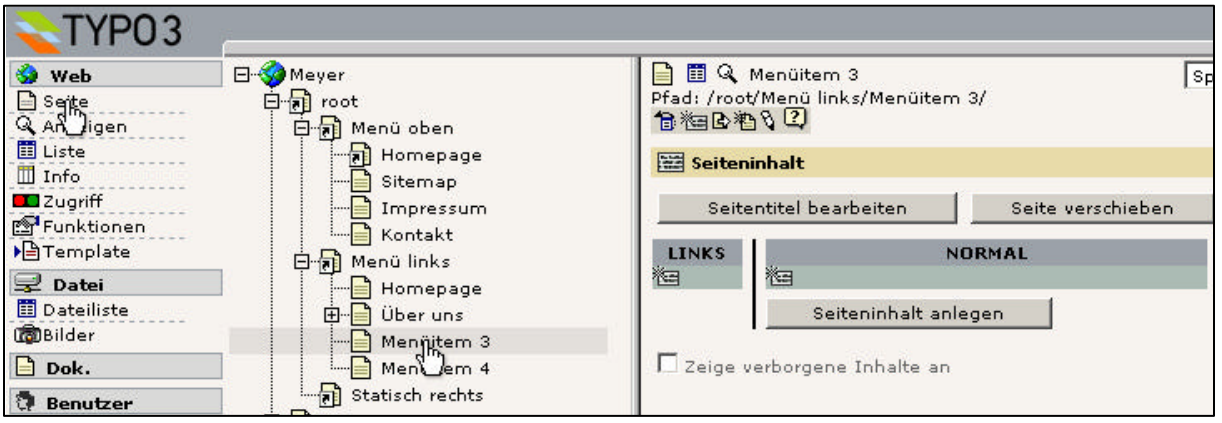
5.3.1.2 Frontend-Plugin: Seiteninhalt / Container anlegen

Wir können nun auf einer beliebigen Webseite unser News-Modul hinzufügen. Eine brauchbare Seite wie z.B. "Aktuelles" gibt es derzeit noch nicht – wir benennen darum die Seite "Menüitem 3" in "Aktuelles" um. Hierzu klicken wir im Seitenbaum auf das Icon, wählen aus dem Popup-Menü den Eintrag "Bearbeite Seiten Header" aus und bearbeiten den Seitentitel in der sich öffnenden Maske auf der rechten Seite.

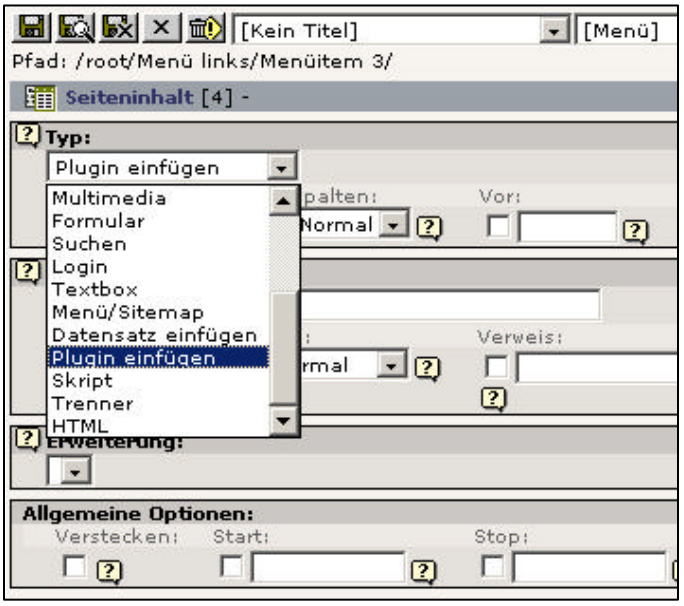


Nachdem wir die Seite mit neuem Seitetitel gespeichert haben, wird uns der neue Seitentitel ebenfalls im Seitenbaum angezeigt.

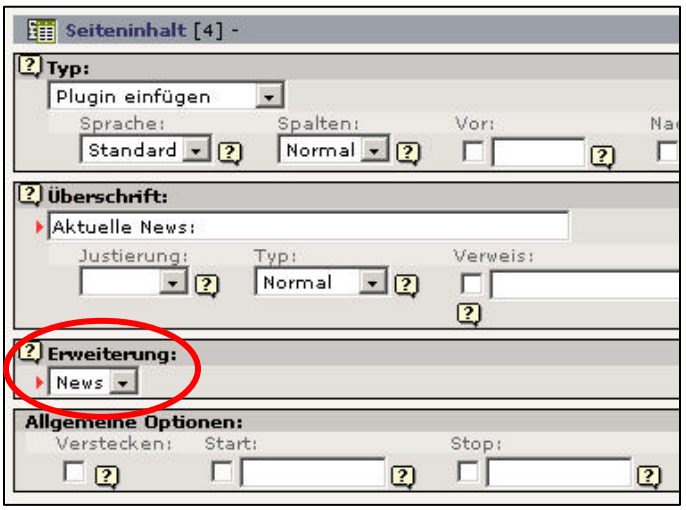
Auf dieser Seite "Aktuelles" können wir nun einen Seiteninhalt ablegen. Hierzu wählen wir im linken Menü den Eintrag "Seite", und im Seitenbaum unsere umbenannte Seite "Aktuelles" aus.



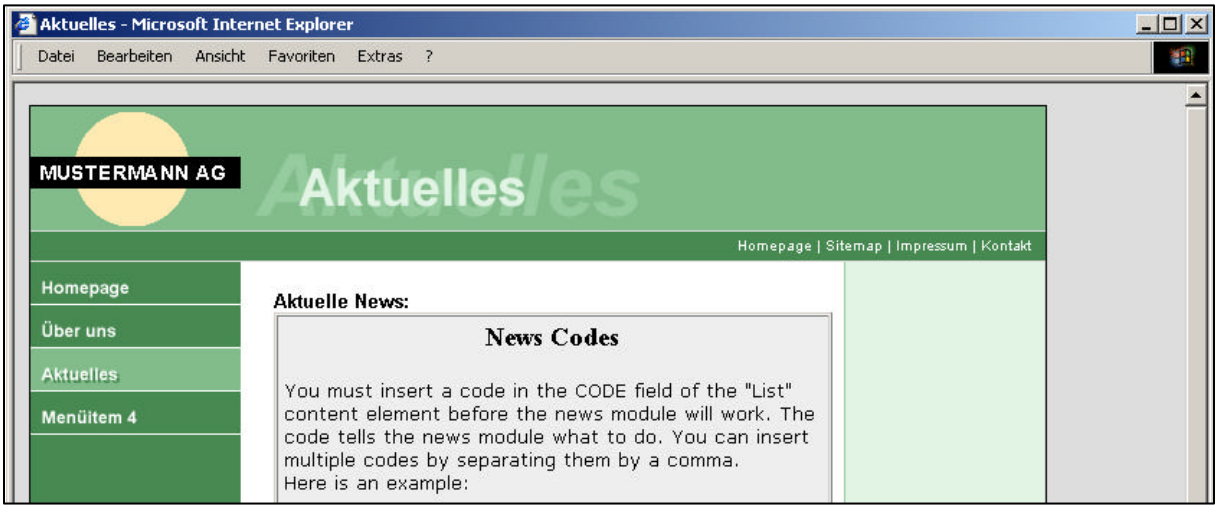
Durch Anklicken des Buttons "Seiteninhalt anlegen" können wir nun einen Seiteninhalt anlegen. Es öffnet sich der "Wizard" für neue Seiteninhalte: Sie können entweder einen ganz normalen Seiteninhalt wie z.B. "Normaler Text" auswählen oder aber im unteren Bereich "Erweiterungen" den Eintrag "Allgemeines Plugin" verwenden. Für welches Element Sie sich auch entscheiden: Sie haben nachträglich immer die Möglichkeit, den Typ des Seiteninhaltes zu ändern, z.B. von "Text" auf "Plugin einfügen". Ändern Sie also den Typ des Seiteninhaltes auf "Plugin einfügen" ab, sofern dieser noch nicht der aktuelle Typ ist.



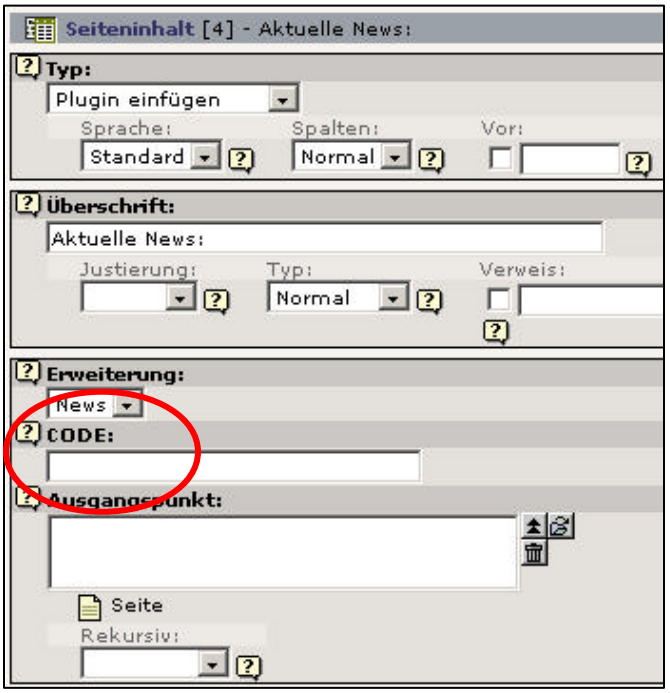
Sie erhalten jetzt eine Maske für Plugins: Sie können unter Anderem einen Inhaltstitel vergeben als auch das gewünschte Frontend-Plugin aus der liste der installierten FE-Plugins auswählen auswählen. In unserem Fall sollte die Liste unter "Erweiterung" nur einen Eintrag "News" enthalten.



Wenn Sie alles richtig angegeben haben und die Seite im Frontend aufrufen, werden Sie eine interessante Meldung erhalten: "News Codes: You must insert a code in the CODE field...".



Diese Meldung gibt an, dass das News-Plugin bereits an der angegebenen Stelle greift. Jedoch benötigt das News-Plugin noch eine weitere Angabe, welche News hier genau dargestellt werden sollen. Dieses wird über ein Feld "CODE" im Seiteninhalt angegeben, welches nach Auswahl, dass das Newsmodul als Erweiterung verwendet werden soll, zur Verfügung steht.



Geben Sie als CODE nun z.B. "LIST" an (bitte beachten Sie, dass der Code immer groß geschrieben werden sollte) und betrachten Sie die Webseite erneut im Frontend. Sie werden feststellen, dass nun keine Meldung mehr erscheint, News-Beiträge aber auch keine angezeigt werden.

Im folgenden Abschnitt 5.3.1.3 finden Sie eine Auflistung der möglichen Codes.

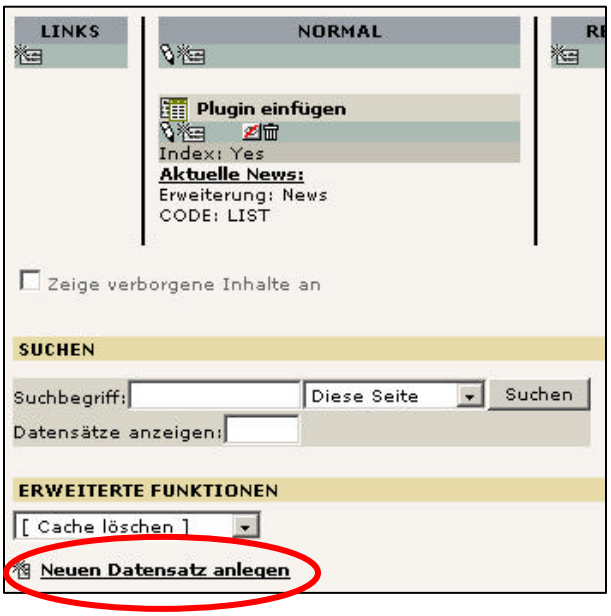
5.3.1.3 Mögliche Codes des News-Moduls

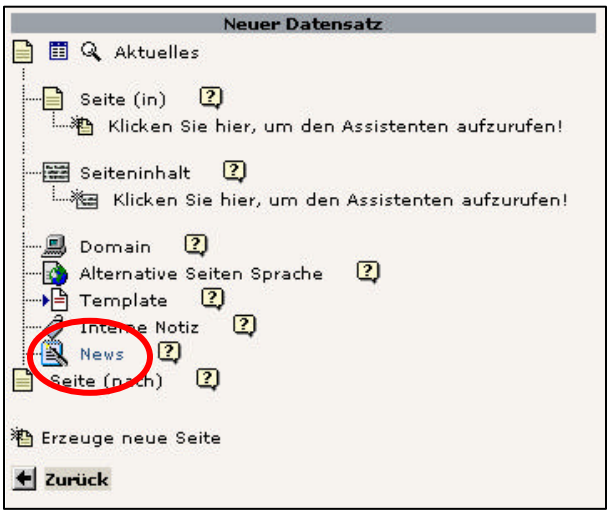
Code	Beschreibung
LATEST	Listet die neuesten, nicht archivierten News aus. Die max. Anzahl an Einträgen kann über die TypoScript-Eigenschaft latestLimit (siehe Referenz weiter unten)
AMENU	Erzeugt ein Menü der archivierten News, die in Zeitspannen unterteilt werden. (Siehe Eigenschaften archiveMode)
LIST	Listet alle News auf. Die Anzahl kann über die Eigenschaften limit beschränkt werden (TypoScript).
SEARCH	Such-Funktion innerhalb der News. Erzeugt ebenfalls die Ergebnis-Seiten.
SINGLE	Zeigt eine bestimmte Nachricht vollständig an.

5.3.1.4 Newsbeiträge erstellen

Wie bereits im Abschnitt 5.3.1.2 gesehen, werden noch keine News-Beiträge angezeigt. Dies dürfte uns auch nicht sonderlich verwundern, wurden doch noch keine News-Beiträge erstellt.

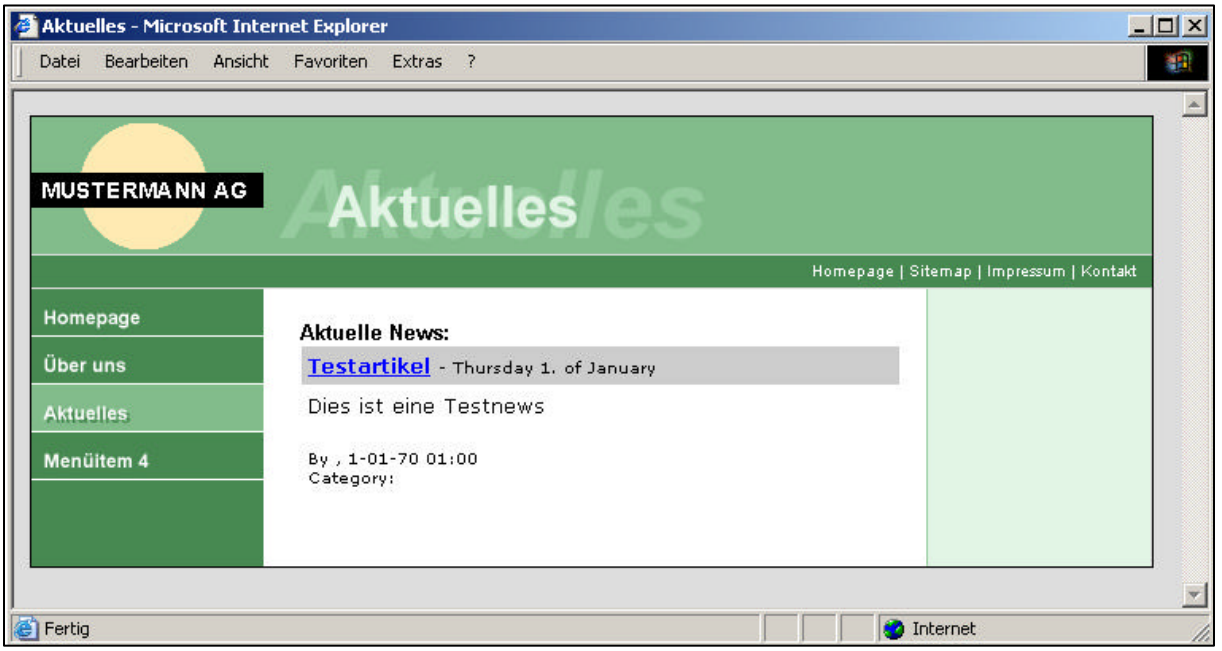
Legen wir nun auf unserer Seite "Aktuelles" einen neuen Datensatz an (keine neue Seite, keinen Seiteninhalt, sondern einen Datensatz), in dem wir auf den Textlink "Neuen Datensatz anlegen" klicken. Wir sehen dann in der Liste der möglichen Datensätze jetzt auch einen Eintrag "News".





Die Maske eines Newsbeitrages ist recht umfangreich und enthält viele Felder, die im Regelfall nicht zwingend benötigt werden. Diese nicht benötigten Felder können den Redakteuren mit Benutzerrechten entzogen werden.

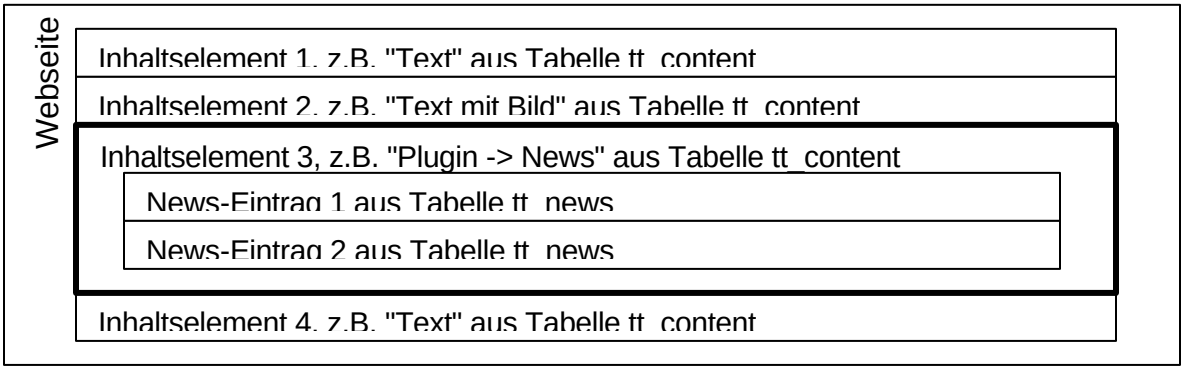
Tragen Sie in die Felder "Titel" und "Text" einen beliebigen Inhalt ein. Die Option "Verstecken" soll deaktiviert werden. Speichern und schließen Sie Ihren News-Datensatz und betrachten Sie das Ergebnis im Frontend:



Wie Sie sehen können, wird dieser Newsartikel bereits in einem bestimmten Design dargestellt. Dieses Design wird in einer Designvorlage definiert. Das News-Modul bringt bereits eine fertige Designvorlage mit, die angepasst werden kann.

5.3.1.5 Frontend-Plugin: Elemente & Container

Das News-Modul ist ein klassisches Beispielt für ein Frontend-Plugin, dass mehrere News-Elemente aufnehmen kann. Das Einfügen eines Seiteninhaltes vom Typo3 "Plugin -> News" stellt einen Container bereit, der, je nach Konfiguration des News-Moduls, mehrere News-Elemente aufnehmen kann. Folgende Skizze soll dieses veranschaulichen:



5.3.1.6 Das News-Modul unter die Lupe genommen

Nehmen wir das News-Modul einmal ein wenig unter die Lupe. Was genau ist überhaupt passiert, als wir das News-Modul installiert haben, wo ist die News-Maske definiert, warum werden im Inhalts-Bereich News angezeigt etc.

Das News-Modul selbst befindet sich im Ordner typo3/ext/tt_news. In diesem Ordner befinden sich einige Dateien, die in folgender Tabelle kurz vorgestellt werden:

Datei	Beschreibung
ext_emconf.php	Hier liegt die generelle Beschreibung des Moduls. Sämtliche Informationen wie z.B. wer das Modul programmiert hat, wie das Modul heißt und was es macht, in welcher Version es vorliegt etc. werden hier gespeichert.
ext_tables.php	Diese Datei erweitert ("extens") bestehende Konfigurationen. Implementierungen im System, z.B. im Clickmenü, das das News-Modul auch unter "Plugin einfügen"->"Erweiterungen" zu finden ist etc.
ext_tables.sql	Hier liegt der SQL-Dump vor. Bei der Installation des Moduls werden nach dieser Datei die Tabellen und Felder angelegt bzw. erweitert, die das News-Modul benötigt.
ext_typoscript_setup.txt	In dieser Datei befindet sich Standart-TypoScript-Code, der inkludiert wird. Der Inhalt läßt sich direkt mit dem Template-Analyser einsehen. Hierdurch wird ermöglicht, das es bereits eine Basis-Konfiguration gibt und das das News-Modul überhaupt als "Inhalt" unterhalb von tt_content implementiert ist. Eine weitere Erläuterung folgt weiter unten.
ext_typoscript_constants.txt	Ähnlich wie ext_typoscript_setup.txt, jedoch Variablen (kein TypoScript). Ebenfalls steht in dieser Datei, wie der Bereich "News" für den Constant-Editor auszusehen hat.
tca.php	Beschreibung dessen, wie die eigentliche News-Maske aussieht. Hier steht die Reihenfolge beschrieben, die Konfiguration jedes einzelnen Feldes (Input-Feld / DateTime-Feld, Trim, Label etc.). Durch Änderungen an dieser Datei können z.B. Module manuell erweitert werden.
locallang_*.php	Diverse Sprachdateien für die Mehrsprachigkeit von Backend-Benutzern.
pi/class.tx_ttnews.php	Eigendliche php-Klasse des News-Moduls, die die Funktionalität für das Frontend bereitstellt.
pi/news_template.tmpl	Mitgelieferte Designvorlage

Wenn das News-Modul über den Erweiterungsmanager installiert wird, laufen intern folgende Integrationen ab:

- Die Datei "ext_tables.sql" wird verarbeitet. Es wird geprüft, ob die Tabellen und Felder vorhanden sind. Alle benötigten Tabellen und Felder werden angelegt.
- In der Datei "typo3conf/localconf.php" wird im Abschnitt $TYPO3_CONF_VARS['EXT']['extList'] das News-Modul zur Liste der installierten Extensions hinzugefügt (tt_news).
- Die temporären Cache-Dateien im Ordner "typo3conf/" werden gelöscht. Diese Dateien beinhalten aus allen installierten Modulen die Konfigurationen von ext_tables.php in aufbereiteter Form.
- Die Dateien ext_typoscript_setup.txt und ext_typoscript_constants.txt werden importiert und mit in die Template-Struktur aufgenommen (einsehbar z.B. mittels Template-Analyser)
- Die Dateien tcap.php sowie pi/class.tx_ttnews.php (und noch weitere Dateien) werden nicht importiert. Sie werden dynamisch zur Laufzeit verarbeitet.

Nun kann es leider immer mal wieder vorkommen, dass die Installation eines Moduls fehlschlägt, der Zugriff auf das Backend z.B. nicht mehr möglich ist. Dieses Problem tritt überlicherweise dann auf, wenn ein Modul für eine andere Typo3-Version bestimmt ist). Hier reicht es im Regelfall aus, folgende Schritte manuell durchzuführen (z.B. mittels FTP):

- Das Modul wird aus der kommaseparierten Liste von $TYPO3_CONF_VARS['EXT']['extList'] aus der Datei "typo3conf/localconf.php" entfernt
- Die temporären Caching-Dateien im Ordner "typo3conf/" werden gelöscht.

Betrachten wir einmal direkt in der Datenbank mittels phpMyAdmin, welche Werte im Datensatz für unseren Inhalts-Datensatz "Plugin Einfügen"->"Erweiterung:News" gespeichert wurden. Hierzu lassen wir uns in phpMyAdmin in der Tabelle "tt_content" die Datensätze anzeigen und sehen uns den Inhalt unseres Datensatzes mit dem Titel "Aktuelle News" genauer an.

Feld	Inhalt	Beschreibung
CType	list	Das Feld "CType" bestimmt, um welchen Content-Typ es sich bei diesem Inhalt handelt (z.B. "Text", "Text mit Bild" etc. Der Content-Typ "Plugin einfügen" wird als Content-Typ "list" gespeichert.
title	Aktuelle News:	
list_type	9	Im Feld "list_type" wird gespeichert, um welches Modul es sich handelt. Da es sich bei dem News-Modul um ein Modul handelt, welches noch aus früheren Typo3-Zeiten ohne Erweiterungsmanager stammt, hat dieses Modul eine "eindeutige" Nummer "9", die in der Datenbank gespeichert wird. Bei neueren Modulen wird hier als Wert der sogenannte "Extension-Key" gespeichert.

Was macht Typo3 nun mit den Daten, wenn versucht wird, den Inhalt darzustellen? In TypoScript haben wir angegeben, dass bei dem Marker "###CONTENT###" Inhalt ausgegeben werden soll. Hierzu steht in TypoScript:

```
[...] marks.CONTENT = CONTENT
[...] marks.CONTENT.table = tt_content
```

Wie bereits im Abschnitt 3.8.2.1 erläutert, sucht Typo3 bei für jeden gefunden Datensatz aus der Tabelle "tt_content" nach einer entsprechenden Darstellungs-Konfiguration in TypoScript. Und auch unser News-Modul bzw. der Container für das News-Modul ("Plugin einfügen") muss sich somit unterhalb von "tt_content" befinden.

Durch das Importieren der Dateien ext_typoscript_setup.txt und ext_typoscript_constants.txt wird vordefinierter TypoScript-Code geladen. Diesen Code können wir uns z.B. im Template-Analyser direkt ansehen, oder aber das Verhalten im Objekt-Browser betrachten. Wo hat sich überall TypoScript-Code verankert, dass z.B. die News als Inhalt angezeigt werden?

Ein Blick in den Objekt-Browser unterhalb von tt_content liefert Aufschluss. Zunächst geht Typo3 in tt_content, um sich die Konfiguration für den aktuellen Datensatz zu suchen. tt_content ist mit "CASE" definiert, unter tt_content.key finden wir, auf welches Datenbankfeld die CASE-Abfrage ausgeführt wird (Abschnitt XXX): CType. Im Feld CType ist bei unserem Datensatz der Wert "list" gespeichert (siehe Tabelle oben). Somit können wir unter tt_content.list fortfahren.

tt_content.list ist COA zugewiesen: An der Position 10 wird die Überschrift geladen (In unserem Fall also die Überschrift für "Aktuelle News:"), Position 20 enthält wiederum eine CASE-Abfrage auf das Feld "list_type" (tt_content.list.20.key). In unserem Datensatz wurde im Feld "list_type" der Wert "9" gespeichert (siehe Tabelle oben). Die nun unter tt_content.list.20.9 erneut zugewiesene CASE-Abfrage mit einer Abfrage auf das Feld "Layout" wird nur für das Layout mit dem Wert "0" definiert. Und an genau dieser Stelle wird

die TypoScript-Definition von "plugin.tt_news" kopiert. Wir müssen nun also an einer anderen Stelle in unserem Objekt-Browser fortfahren, um das Verhalten zu untersuchen.

Unter "plugin.tt_news" sehen wir eine Zuweisung des Objektes "USER". Dieses Objekt erfordert zwingend die Angabe einer php-Klasse mit Funktion. In "plugin.tt_news.userFunc" können wir eine Zuweisung "tx_ttnews->main_news" erkennen: Dies gibt an, dass aus der Klasse "tx_ttnews" die Funktion "main_news" aufgerufen werden soll.

Doch woher weiß Typo3 nun, wo sich die Klasse "tx_ttnews" befindet? Einzubindende php-Klassen werden unterhalb von "includeLibs" angegeben. So können wir z.B. unter "includeLibs.ts_ttnews" eine Zuweisung "EXT:tt_news/pi/class.tx_ttnews.php" erkennen. Der in der Zuweisung angegebene Bezeichner "EXT:" gibt an, dass eines der globalen oder localen Extension-Ordner (typo3/ext oder typo3conf/ext) verwendet werden soll.

Warum "ts_ttnews" statt "tx_ttnews"? Hier gibt es eine ganz einfach Erklärung: Es ist ganz gleich, was hier genau angegeben wird. Es könnte hier auch "includeLibs.1 = EXT..." stehen. Wesendlich ist, dass mehrere Klassen eingebunden werden können und hierzu eine weitere, eindeutige Separation unterhalb von "includeLibs" notwendig ist.

5.3.1.7 Kapselung Funktionalität, Konfiguration und Design

Die Funktionalität steht bei Modulen im Regelfall im Mittelpunkt und wird durch die php-Klasse zur Verfügung gestellt. Hier steht drin, was das Modul genau machen soll. Theoretisch ist es möglich, in dieser php-Klasse auch direkt "hardcodiert" das Design des Frontends zu verankern oder bestimmte Eigenschaften (Datenbankzugriff etc.) direkt zu setzen. Da dieses jedoch häufig zu einem großen, nachträglichen Mehraufwand führt und eine Portierung nicht mehr problemlos währe (Grundgedanke: "Install in one click"), werden Frontend-Module, wie z.B. das News-Modul, in drei Bereiche unterteilt: Funktionalität (php-Klasse), Konfiguration (TypoScript) und Design (Designvorlage).

Funktionalitäten können konfiguriert, also auf die jeweiligen Bedürfnisse angepasst werden. Beim News-Modul wurde z.B. im Seiteninhalt ("Plugin einfügen") ein Code angegeben, "was das News-Modul machen soll". Wir haben dort als Code beispielhaft "LIST" angegeben. Eine andere mögliche Code-Konfiguration währe z.B. "LATEST" gewesen. Es werden ebenfalls Funktionalitäten wie z.B. alternierende Hintergrundfarben, Beschränkung auf eine bestimmte Anzahl von angezeigten Datensätzen zur Verfügung gestellt. Auch der Dateiname der gekapselte Designvorlage ist konfigurierbar.
Die Konfiguration wird im Regelfall über TypoScript vorgenommen.

5.3.1.8 Das News-Modul konfigurieren

Wie unter 5.3.1.7 erwähnt, sind Konfigurationen des News-Moduls über TypoScript möglich. Hierbei handelt es sich im Regelfall nicht direkt um das eigendliche TypoScript: Vielmehr ist es ein Mischmasch aus von der php-Klasse erwarteten bzw. ermöglichten Variablen und tatsächlichem TypoScript, der ebenfalls von der php-Klasse verarbeitet wird. Leider ist dieses nicht direkt erkennbar.

Wir haben im Abschnitt 5.3.1.6 gerlernt, dass die php-Funktion in plugin.tt_news.userFunc angegeben wurde:

```
plugin.tt_news = USER
plugin.tt_news {
   userFunc = tx_ttnews->main_news
   templateFile = typo3/ext/tt_news/pi/news_template.tmpl
   [...]
}
```

Das USER-Objekt erwartet aber nur eine Eigenschaft "userFunc". Sämtliche zusätzliche Eigenschaften werden als "Variablen" in einem großen Array an die unter "userFunc" angegebene php-Klasse übergeben und können von dieser verarbeitet werden.

Die Schreibweise dieser Variablen obliegt dem Autor des Moduls. Autoren sollten sich an die Regeln halten, wie Eigenschaften in TypoScript geschrieben werden: Das erste Eigenwort wird klein geschrieben, alle folgenden Eigenschaften groß. Die Eigenschaft bzw. Variable "templateFile" ist also mit diesen Konventionen konform. Dies muss allerdings nicht so sein.

Folgende Tabelle gibt eine Übersicht, welche Konfigurationen zur Verfügung stehen.

Eigenschaft	Datentyp / Erwarteter Wert	Beschreibung
Konfigurationsmöglichkeiten des News-Moduls		
templateFile	Dateiname	Die Datei mit der Designvorlage, die Marker enthält. Hier kann das grundlegende Design als auch die Sprache der News geändert werden. Ein vollständiges englisches Beispiel ist bereits vordefiniert.
alternatingLayouts	Integer	Unter alternierenden Layouts wird verstanden, das sich Farben etc. von Eintrag zu Eintrag ändern können, um sich somit insbesondere von Ihren direkten Nachbarn zu unterscheiden. Häufige Anwendung ist eine sich abwechselnde Hintergrundfarbe Der Wert alternating_Layouts = 1 bedeutet, das keine (!) alternierenden Layouts verwendet werden. Wenn der Wert auf 2 gesetzt wird,

		können sich die Farben nun z.B. abwechseln. So wird z.B. der Subpart ###NEWS_1### jedes zweite mal verwendet, also abwechselnd mit dem Subpart ###NEWS###.
pid_list	String	Die eindeutige Seitennummer (pid) der Seite, von der die News geholt werden sollen. Mehrere Angaben erfolgen kommasepariert. Kann alternativ im Plugin (also dem Seiteninhalt, auf dem auch die CODES angegeben werden) gesetzt werden.
code	string	Der CODE, der sagt, was das Script zu machen hat. Wird in der Regel direkt bei den Eigenschaften des Seiteninhaltes (Plugin) angegeben.
recursive	int	Anzahl der rekursiven Ebenen, die von den Einträgen innerhalb von pid_list mit nach Nachrichten durchsucht werden sollen.
PIDitemDisplay	Integer	Die eindeutige Seitennummer (pid) der Seite, die aufgerufen werden soll, wenn eine Nachricht in seiner Gesamtheit aufgerufen wird. Die Zielseite muss ebenfalls die Plugin-Erweiterung „News“ mit dem Code „SINGLE“ enthalten.
PIDsearch	Integer	Gleiches wie oben, jedoch für die Suchanfrage. Die Zielseite muss ebenfalls die Plugin-Erweiterung „News“ mit dem Code „SEARCH“ enthalten.“.
backPid	Integer	Die Seitennummer (pid) die aufgerufen werden soll, wenn nach dem Betrachten einer Nachricht in seiner Gesamtheit (code = SINGLE) auf den Textlink „Zurück“ geklickt wird.
limit	Integer	Die maximale Anzahl an Nachrichten, die auf einer Seite (insbesondere bei dem code = LIST) angezeigt werden.
latestLimit	Integer	Wie oben, jedoch für den code = LATEST
enableArchiveDate	Boolean 1/0	Aktiviert den Archivierungsmodus.
datetimeDaysToArchive	Integer	Gibt die Anzahl der Tage an, ab denen eine Nachricht archiviert wird. Um diese Eigenschaft nutzen zu können, muss enableArchiveDate = 1 gesetzt sein (der Archivierungsmodus als aktiviert sein).

archiveMode	string	Der Archivierungs-Modus gibt an, in welchen Zeitintervallen bzw. Perioden archivierte News ausgegeben bzw. zusammengefasst werden sollen. Mögliche Werte sind: • month (monatlich) • quarter (quartalsweise) • year (jährlich)
archiveMenuNoEmpty	Boolean 1/0	Wenn diese Eigenschaft = 1 gesetzt ist, enthält keine Periode leere Einträge
date_stdWrap	String	Gibt an, wie die Anzeige des Datums erfolgen soll. Beispiel: plugin.tt_news.date_stdWrap.strftime = %e-%m-%y
time_stdWrap	String	Wie oben, jedoch für die Zeit. Beispiel: plugin.tt_news.time_stdWrap.strftime = %H:%M:%S
title_stdWrap author_stdWrap email_stdWrap caption_stdWrap subheader_stdWrap content_srdWrap links_stdWrap related_stdWrap category_stdWrap	String	Die Wraps für die einzelnen Teilbereiche.
imageCount	Integer	Hier wird angegeben, wie viele Bilder maximale je Nachricht angezeigt werden. Default ist = 1
displayLatest.image.file.maxW displayList.image.file.maxW displaySingle.image.file.maxW	Integer	Hier kann angegeben werden, welche Breite die Bilder in den jeweiligen Teilgebieten haben sollen.

5.3.1.9 Die Designvorlage anpassen

Das Design kann relativ problemlos an die eigenen Bedürfnisse angepasst werden. Wichtig ist allerdings, dass sich die Designvorlage nicht im globalen Extension-Ordner befindet, da sie ansonsten bei einem Update durch die Default-Version überschrieben werden würden.

Kopieren Sie sich die Designvorlage am Besten in den Ordner "fileadmin/" oder einen Unterordner von diesem. Passen Sie anschließend in TypoScript diesen Ort der Designvorlage an:

```
plugin.tt_news.templateFile = fileadmin/template_news.tmpl
```

Die Dateierweiterung einer Designvorlage muss nicht "tmpl" sein. Diese Erweiterung soll lediglich aufzeigen, dass es sich bei dieser Datei um eine Designvorlage handelt. Möglich sind aber auch Erweiterungen wie z.B. "html", "txt" etc.

Der Inhalt dieser Designvorlage setzt sich aus vielen verschachtelten Teilbereichen (Subparts) zusammen. Verwenden Sie am Besten die Original-Designvorlage als Basis und passen Sie diese Ihren Bedürfnissen an. Sie können problemlos Marker entfernen. Teilbereiche sollten aber bestehen bleiben.

5.3.1.10 Statische Darstellung von News

Mit folgendem TypoScript-Code ist es z.B. möglich, das auf der rechten Seite permanent die News angezeigt werden:

```
marks.RECHTS < plugin.tt_news
marks.RECHTS {
   defaultCode = LATEST
   code >
   code = LATEST
   latestLimit >
   latestLimit = 10
   pid_list >
   pid_list = 67
}
```

Leider ist es mit der gelieferten Version des News-Moduls nicht möglich, mehrere News-Bereiche auf einer Seite einzusetzen. So ist z.B. häufig gewünscht, das auf der rechten Seite eine „Mini-Voransicht“ der News existiert (z.B. Latest News). Wenn dann eine solche Nachricht ausgewählt wird, der Benutzer also praktisch einen Artikel in seiner vollen Länge lesen möchte, dann würde der Artikel ebenfalls in seiner vollen Länge im rechten Fenster der Mini-Ansicht angezeigt und, wie auch gewünscht, im Content-Bereich. Einzigste Abhilfe, die das Newsmodul in seiner Form bietet, ist, dass der rechte Bereich auf den Detail-Seiten (Single) ausgeblendet wird.

Kapitel 6: Diverses

6.1 Anpassungen mittels Conditions

Conditions (Bedingungen) können dazu verwendet werden, um Eigenschaften unter bestimmten Umständen zu setzen. "Wenn etwas bestimmtes gegeben ist, dann soll die Eigenschaft xyz einen Wert abc erhalten".

Eine vollständige Übersicht über die Möglichkeiten von Conditions finden Sie im Kapitel 8.7. In diesem Kapitel soll vorwiegend die Anwendung von Conditions vorgestellt werden.

Zur beispielhaften Darstellung von Conditions soll je nach Uhrzeit ein Inhalt "Guten Morgen", "Herzlich Willkommen" oder "Guten Abend" ausgegeben werden:

```
seite.40 = TEXT
[hour = > 2]
seite.40.value = Guten Morgen
[hour = > 11]
seite.40.value = Herzlich Willkommen
[hour = > 19]
seite.40.value = Guten Abend
[global]
```

Conditions dürfen sich nicht (!) innerhalb von geschweiften Klammern befinden! Sie müssen also sicherstellen, dass, vor der Anwendung von Conditions, alle geschweiften Klammern geschlossen werden.

Conditions werden immer mit einem abschließenden [global] beendet. Nach dem [global] wird TypoScript wieder ohne Berücksichtigung auf besondere Gegebenheiten geparsed.

Die Syntax von Conditions ist wie folgt:

```
[Gegebenheit1 = Wert1] [Gegebenheit2 = Wert2]
   TypoScript_A
[else]
   TypoScript_B
[global]
```

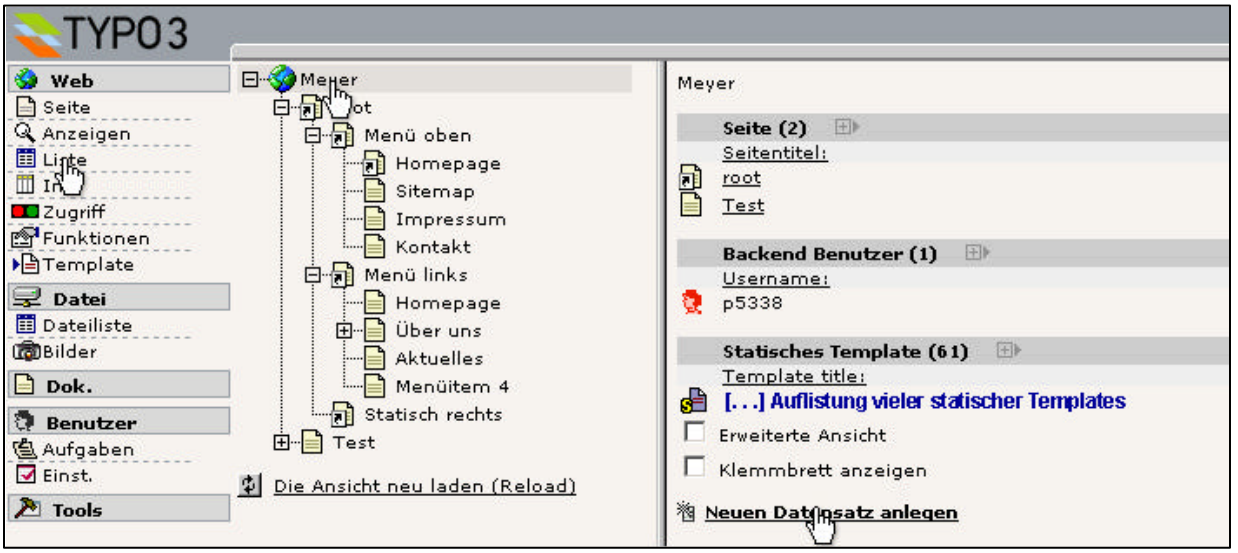
Wenn Gegebenheit1 oder / und Gegebenheit2 zutreffen, wird der Bereich "TypoScript_A" ausgeführt, ansonsten der Bereich "TypoScript_B".

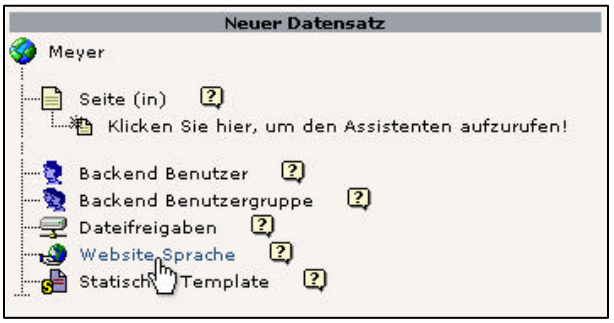
6.2 Mehrsprachige Webseiten

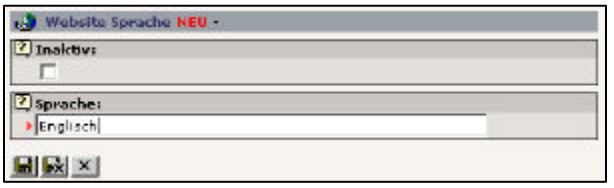
Mehrsprachige Internet-Präsentationen werden von Typo3 nativ unterstützt. Hier gibt es zwei Möglichkeiten, wie Internetpräsentationen mehrsprachig erstellt werden können: Zum Einen durch zwei (oder mehrere) unterschiedliche Seitenbäume und zum Anderen durch einen Seitenbaum. In diesem Abschnitt soll vorgestellt werden, wie Präsentationen mit einem Seitenbaum mehrsprachig erstellt werden können.

6.2.1 Seitensprachen, Seiten und Inhalte übersetzen

Zunächst müssen wir in Typo3 einer weitere Sprache (zusätzlich zur bereits vorhandenen Default-Sprache) anlegen. Hierzu klicken wir auf im linken Menüeintrag auf "Liste", wählen als aktuelle Seite die Rootebene aus (Weltkugel, nicht die Seite "root"!) und legen Sie einen neuen Datensatz vom Typ "Website-Sprache" an.

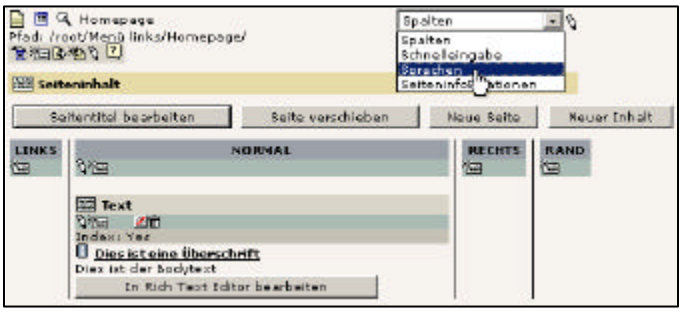






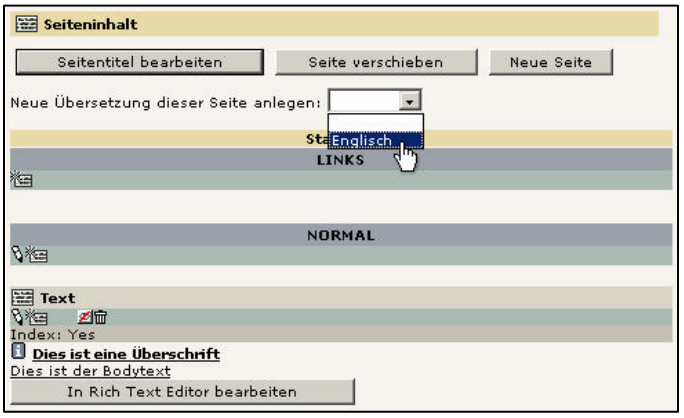
Geben Sie als Sprache z.B. "Englisch" an und speichern Sie Ihren neuen Datensatz.

Nun haben wir eine neue Website-Sprache angelegt und müssen nun unsere Seiten und deren Inhalte entsprechend übersetzten. Hierzu gehen wir wieder in den Bereich "Seite" und wählen beispielsweise unsere Seite "Homepage" aus, auf der sich bereits Inhalt befindet. Wir können nun im oberen, rechten Bereich aus der Auswahlbox einen Menüeintrag "Sprachen" auswählen:

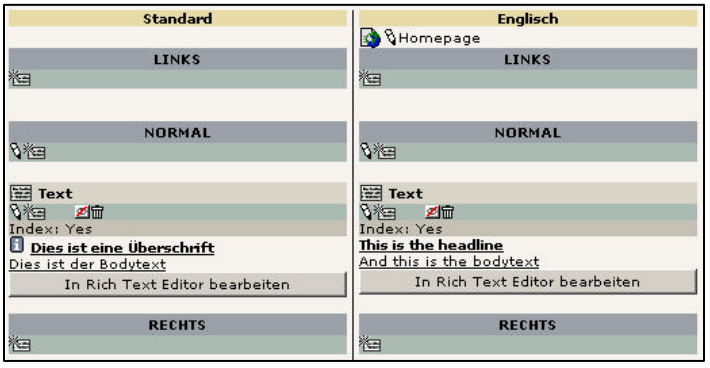


Das Element "Sprachen" steht nur dann zur Verfügung, wenn auch eine zusätzliche Website-Sprache angelegt wurde!

Wir können nun von unserer aktuellen Seite eine Übersetzung anlegen:



Geben Sie nun auf der folgenden erscheinenden Maske eine Übersetzung des Seitentitels an (Die Übersetzung von "Homepage" sollte ebenfall "Homepage" sein). Achten Sie darauf, dass die übersetzte Version nicht als "Versteckt" gekennzeichnet ist und speichern Sie Ihre Übersetzung des Seitentitels. Legen Sie in der Spalte "Englisch" nun eine Übersetzung des Seiteninhalt "Dies ist eine Überschrift" vom Typ "Text" an und speichern Sie diesen neuen Seiteninhalt. Die Darstellung im Backend sollte nun wie folgt aussehen:



6.2.2 TypoScript und Mehrsprachigkeit

Wenn wir unsere Seite nun im Frontend betrachten, werden wir feststellen, dass unsere Seite nun sowohl den englischen als auch den deutschen Seiteninhalt enthält. Dieses dürfte nur in den seltesten Fällen so gewünscht sein. Gewünscht ist z.B., dass nur der deutsche Seiteninhalt angezeigt wird, bis eine andere Sprache ausgewählt wird.

Damit nun nur die Seiteninhalte der deutsche Sprache dargestellt werden, müssen wir in TypoScript unsere Konfiguration für den Marker "CONTENT" erweitern:

```
seite.10.marks.CONTENT {
   table = tt_content
   select.orderBy = sorting
   select.where = colPos = 0
   select.languageField = sys_language_uid
}
```

Ein erneutes Betrachten der Webseite im Frontend zeigt uns, dass jetzt nur noch der deutsche Inhalt dargestellt wird. Wir müssen es nun allerdings noch ermöglichen, den englischen Inhalt ausgeben zu lassen. Hierzu erweitern wir unser Template um weitere folgende Zeilen, direkt im oberen Abschnitt, möglich oberhalb von "seite = PAGE":

```
01    config.linkVars = sprache
02    config.sys_language_uid = 0
03    config.language = de
04    [globalVar = GP:sprache=1]
05       config.sys_language_uid = 1
06       config.language = en
07    [global]
```

- In Zeile 1 wird angegeben, dass ein Parameter "sprache" mit jedem von Typo3 aus erzeugten Link weitergegeben wird.
- In den Zeilen 2 und 3 wird die ID der Sprache auf "0" (Defaultsprache), und der "Language-Label" auf "de" gesetzt.
- In Zeile 4 folgt eine Condition-Bestimmung: Die Zeilen 5 und 6 werden nur dann ausgeführt, wenn der übergebene Parameter "sprache" gleich 1 ist.
- In Zeile 7 wird der Condition-Bereich verlassen.

Sehen wir uns die Seite im Frontend erneut an und erweitern unsere URL nun um den Parameter "&sprache=0", also z.B. auf "index.php?id=18&sprache=0". Wir werden die Seite erneut in unserer Default-Sprache "Deutsch" betrachten können. Ändern wir den Parameter allerdings auf "&sprache=1" um, so wird die Seite in der übersetzten englischen Version erscheinen.

Ebenfalls werden sämtliche Links, so z.B. das Menü, jeweils mit dem gültigen Parameter "&sprache" versehen. Hierfür ist die Konfiguration "config.linkVars = sprache" verantwortlich.

Im Regelfall wird als Parameter "&L=" verwendet. Es gibt fertige Frontend-Module, die eine Mehrsprachigkeit nur in Kombination mit dem Parameter "&L=" ermöglichen. Diese Module wurde nicht "sauber" entwickelt. Um solche Probleme zu vermeiden, gewöhnen Sie sich als Parameter für die Mehrsprachigkeit "&L=" an. Die Darstellung mit dem Parameter "&sprache=" diente ausschließlich anschaulichen Zwecken.

Mit dem Parameter "&L=" sieht demnach die Konfiguration wie folgt aus:

```
01     config.linkVars = L
02     config.sys_language_uid = 0
03     config.language = de
04     [globalVar = GP:L=1]
05        config.sys_language_uid = 1
06        config.language = en
07     [global]
```

Um jetzt noch einen komfortablen Sprachwechsel zu ermöglichen, müssen wir den Parameter übergeben können.

```
seite.20 = TEXT
seite.20.wrap = <a href="index.php?id=|&L=1>Englisch</a>
seite.20.field = uid
[globalVar = GP:L=1]
   seite.20.wrap = <a href="index.php?id=|&L=0>Deutsch</a>
[global]
```

Selbstverständlich kann obiger "Sprachwechsler" auch auf einen Marker gelegt werden, verfeinert mit Grafiken etc.

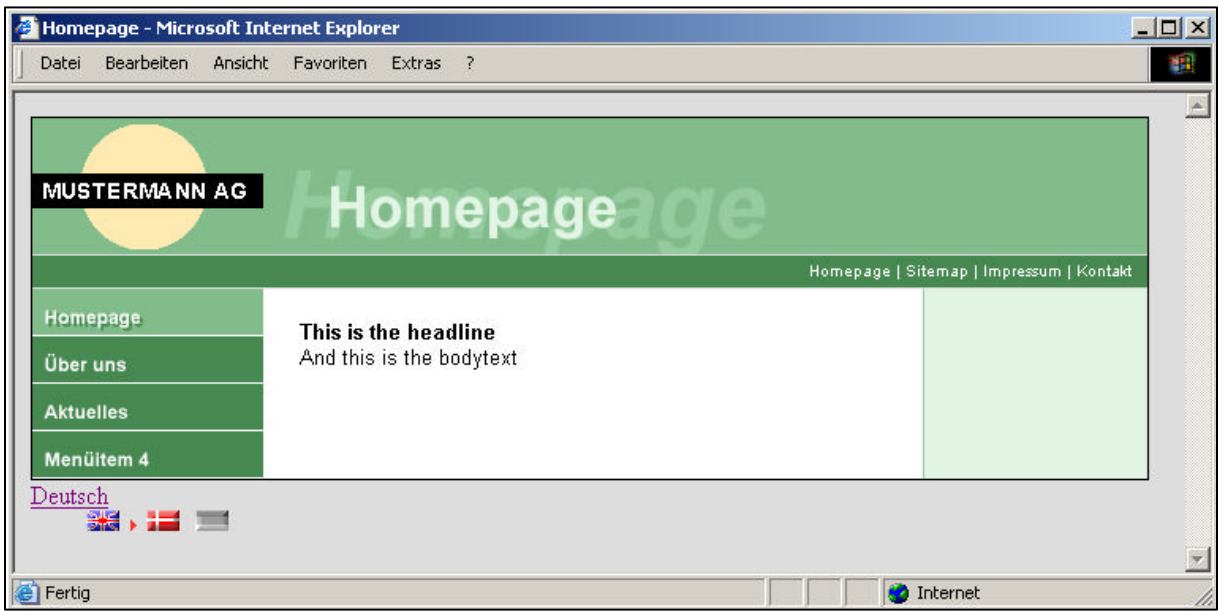
Es gibt ein interessantes Script, mit dem ein funktionsreicher, grafischer Sprachwechsler integriert werden kann. Dieses Script kann recht einfach eingebunden werden:

```
seite.30 = PHP_SCRIPT
seite.30.file = media/scripts/example_languageMenu.php
```

Dieser Sprachwechsler erwartet in der Grundversion den Parameter "&L=" als Sprachparameter.

Die genaue Konfiguration, welche Sprache sich hinter welcher ID befindet, muss direkt im Source-Code dieses Scriptes geändert werden. Um kleine "mini"-php-Kenntnisse werden Sie bei der Anpassung dieses Moduls nicht herumkommen.

Die wie oben konfigurierte Seite sollte bei "&L=1" wie folgt aussehen:



6.3 Druckerfreundliche Version

Häufiger Anwendungsfall bei Internetpräsentationen ist eine spzielle Druckversion, die im Regelfall den gleichen Inhalt hat, wie die "normale" Präsentation, jedoch ein anderes Design. Wir wollen hier nun beispielhaft eine Druckversion realisieren.

Im Abschnitt 3.1.1 haben wir bereits die Eigenschaft "typeNum" des PAGE-Objektes kennengelernt. Und genau diese typeNum können wir uns nun für die Druckversion zunutze machen. Wir erweitern unser Template um folgende Zeilen:

```
druckversion = PAGE
druckversion {
   typeNum = 1
   10 = CONTENT
   10.table = tt_content
   10.select.orderBy = sorting
   10.select.colPos = 0
}
```

Wenn wir nun unsere Seite im Frontend aufrufen, sehen wir noch immer das klassische Design. Um nun unsere gerade erstellte Druckversion betrachten zu können, müssen wir die URL noch um einen Parameter "&type=1" erweitern:

```
index.php?id=18&type=1
```

Wir können jedoch erkennen, dass wieder beide Sprachen dargestellt werden. Um nur die jeweils gültige Sprache anzeigen zu lassen, müssen wir den Content-Bereich unserer Druckversion noch um diese zusätzliche Funktion erweitern:

```
druckversion = PAGE
druckversion {
   typeNum = 1
   10 = CONTENT
   10.table = tt_content
   10.select.orderBy = sorting
   10.select.colPos = 0
   10.select.languageField = sys_language_uid
}
```

Selbstverständlich kann für die Druckversion auch eine Designvorlage verwendet werden. Hierzu ist, wie im Kapitel 3 beschrieben, das Objekt TEMPLATE zu verwenden, also z.B.

```
druckversion.10 = TEMPLATE
druckversion.10 {
     template = FILE
     template.file = fileadmin/druckversion.html
     marks.INHALT = CONTENT
     [...]
}
```

Es wäre nun allerdings eine Zumutung, dem Benutzer einer Webseite sagen zu müssen, dass manuell ein Parameter "&type=1" angefügt werden muss, um eine Druckansicht der aktuellen Seite zu erhalten. Dieses läßt sich auch eleganter lösen.

Erweitern wir unser Template erneut um ein Element "seite.50":

```
seite.50 = TEXT
   wrap = <a href="index.php?id=|&type=1">Druckversion</a>
   field = uid
}
```

6.3.1 Zusätzliche Parameter mitreichen (Sprache)

Diese Form der Druckversion ist funktionsfähig, jedoch werden wir merken, dass spätestens nach einer Sprachumschaltung die Druckversion nur die Defaultsprache anzeigt. Wir müssen nun also noch den GET-Parameter "L" mitgeben:

```
seite.50 = COA
seite.50.10 = TEXT
seite.50.10.wrap = <a href="index.php?id=|&type=1
seite.50.10.field = uid
seite.50.20 = TEXT
seite.50.20.wrap = &L=|">Druckversion</a>
seite.50.20.data = GPvar:L
```

6.3.2 Besondere Darstellung von tt_content

Nun kann es immer wieder vorkommen, dass die Definition von tt_content (Darstellung des Seiteninhaltes) für die Druckversion nicht besonders gut geeignet ist. tt_content wird jedoch auch bei der Darstellung des Inhaltes für die Druckversion benötigt. Es werden also zwei Konfigurationen für tt_content benötigt, je nach "normaler" Darstellung oder Druckversion.

Dieses kann mittels Conditions erreicht werden. Erweitern wir nun unsere Darstellungs-Konfiguration für die Überschrift, die wir im Abschnitt 4.3.11.6 vorgenommen haben um eine Funktionalität, dass für die Druckversion die Schriftgröße "3" sein soll:

```
lib.stdheader >
lib.stdheader = CASE
lib.stdheader {
   key.field = header_layout
   1 = TEXT
   1.field = header
   1.wrap = <font face="Arial" size="2"><b> | </b></font><br>
   default < .1
}
[globalVar = GP:type=1]
   lib.stdheader.1.wrap (
      <font face="Arial" size="3"><b> | </b></font><br>
   )
   lib.stdheader.default < .1
[global]
```

Bitte beachten Sie, dass Conditions sich nicht(!) innerhalb von geschweiften Klammern befinden dürfen! Bei der Überschreibung des wraps wurde eine runde Klammer verwendet, damit kein unbeabsichtigter Umbruch in dieser Dokumentation zu Verwirrung sorgt.

6.4 Suchmaschinenfreundliche URLs

Suchmaschinen wie google durchlaufen häufig keine Internetseiten mit URL's der Art ".php?". Seiten, bei denen Parameter enthalten sind, werden nicht weiter indiziert.

Es gilt nun, diese Internetseiten so darstellen zu lassen, dass kein Fragezeichen mehr enthalten ist und Parameter als solche nicht mehr erkennbar sind.

Es gilt nun das Problem der kryptischen Namensgebung anzugehen: Wie macht man aus index.php?id=18 z.B. die Seite homepage.html. Hier hat Typo3 vorgesorgt und bietet mächtige Eigenschaften: Mit SimulateStaticDocuments.

6.4.1 Den Webserver vorbereiten

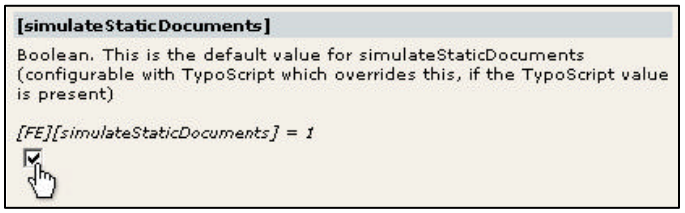
Bei Kunden von mittwaldmedien CM Services sind die in diesem Abschnitt genannten Einstellungen bereits voreingestellt.

Um in den Genuss von SimulateStaticDocuments zu kommen, muss der Webserver zunächst vorbereitet werden. So ist es z.B. zwingend erforderlich, das eine .htaccess-Datei auf dem Webserver abgelegt wird. Diese .htaccess-Datei muss folgenden Inhalt haben:

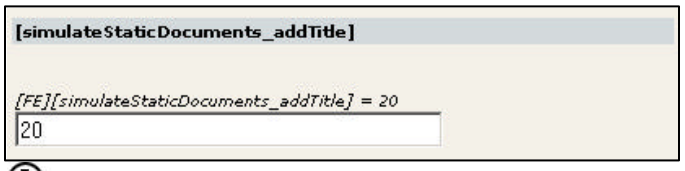
```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule   ^[^/]*\.html$  index.php
```

Hierdurch wird eine an die existente Internetseiten mit der Endung ".html" automatisch der Seite index.php zugewiesen.

Wir müssen Typo3 selbst noch beibringen, das es Seiten in der Form homepage.html erzeugen soll (statt index.php?id=18). Im Install-Tool können wir unter "AllConfiguration" im Abschnitt "FE" diese Option aktivieren:



Wir sollten ebenfalls überprüfen, ob für die Konfigurationsvariable "simulateStaticDocuments_addTitle" ein positiver Wert angegeben wurde (z.B. "20"):

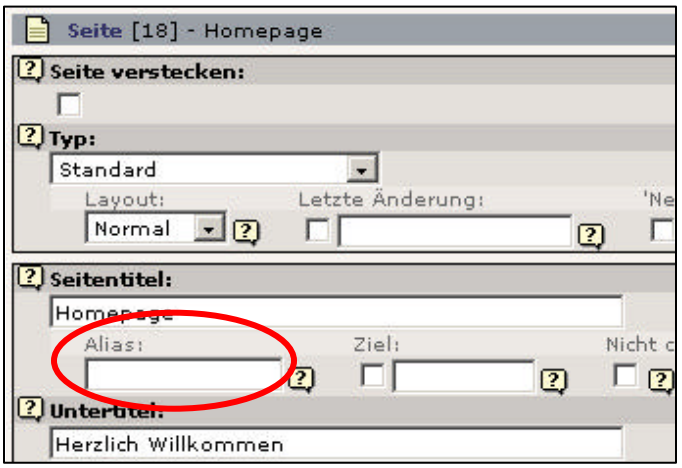


Diese Änderungen werden, wie alle Änderungen, die im Install-Tool getätigt werden, in der Datei localconf.php gespeichert.

Betrachten wir nun unsere Seiten im Frontend erneut, so werden die Seiten nicht mehr mit der URL "index.php?id=18" dargestellt, sondern mit "18.0.html"

6.4.2 Mit Alias-Namen arbeiten

In den Seiteneigenschaften haben wir die Möglichkeit, einen Alias-Namen anzugeben. Sollte dieses Feld Alias nicht vorhanden sein, so empfiehlt sich, die Option „Zweite Optionspalette anzeigen“ ganz unten auf der Seite zu aktiveren. Geben Sie nun in diesem Feld z.B. den Alias-Namen "homepage" ein.



Betrachten wir uns die Seiten erneut im Frontend (reload) und wählen im Menü den Menüpunkt "Homepage" aus, so können wir mit Freude feststellen, dass die URL "homepage.0.html" lautet.

Die angegebene 0 ist jedoch noch etwas störend. Sie gibt an, das die TypeNum 0 ist. Der Grund für den Einsatz von TypeNums wurde bereits weiter oben beschrieben. Sofern die TypeNum aber = 0 ist (default), kann die explizite Übermittlung einer TypeNum deaktiviert werden. Hierzu ist in unserem Template (TypoScript) folgende Zeile einzufügen:

```
config.simulateStaticDocuments_noTypeIfNoTitle = 1
```

Hierdurch wird die Null für TypeNum = 0 weggelassen. Das Ergebnis unserer Änderungen sieht nun wie folgt aus:

```
http://www.meinedomain.de/homepage.html
```

6.5 Benutzerrechte Backend-Redakteure

Typo3 hat eine mächtige Rechteverwaltung inklusive, die auf dem ersten Blick nicht ins Auge fällt. Grund hierfür ist, das bei einer Standard-Installation von Typo3 ein Admin-User angelegt wurde. Ebenfalls werden neue Benutzer oftmals auch als Admin-User angelegt. Mit Admin-Rechten ist die gesamte Struktur des Rechtekonzeptes nicht erkennbar.

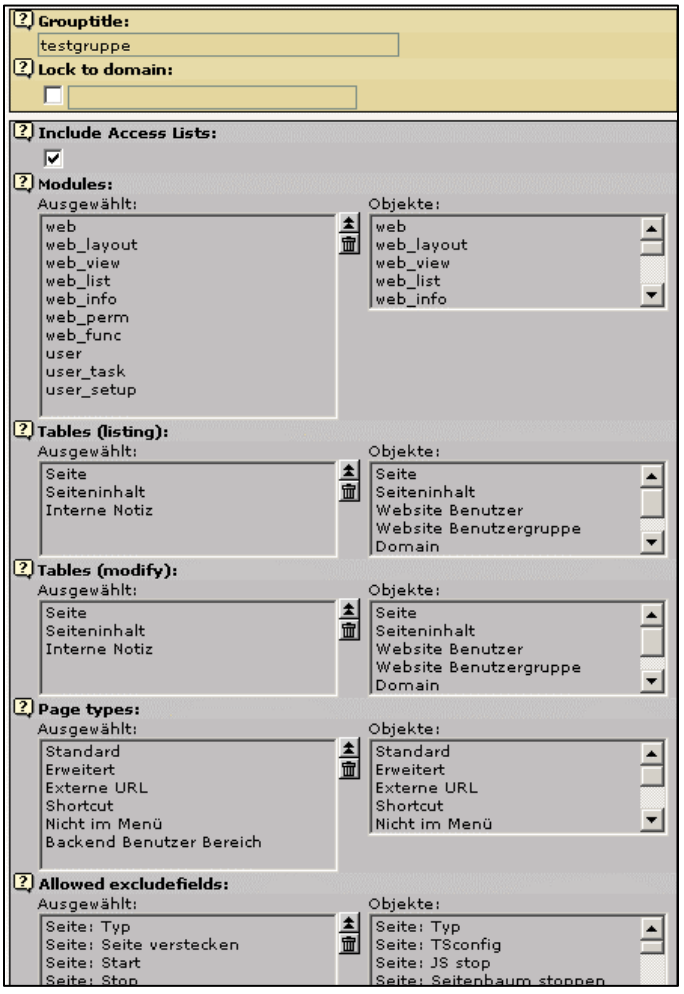
Das Rechtekonzept basiert auf Benutzergruppen und den Benutzern selbst. Ein Benutzer kann Mitglied mehrerer Gruppen sein, die Rechte addieren sich. Hierbei arbeitet Typo3 mit sogenannten Positivlisten. Nur wenn eine Rechteeigenschaft explizit gesetzt wurde, stehen diese Rechte dem Benutzer auch zur Verfügung. Ausnahme sind hier die Admin-User. Für diese Admin-User wird keine Positivliste benötigt. Sie haben und dürfen grundsätzlich alles.

Wichtig zu wissen ist ebenfalls, das für eine richtige Benutzerverwaltung an drei Stellen gearbeitet werden muss: Es muss eine Benutzergruppe geben, bei der in der Access-Liste die durchdachten Eigenschaften aktiviert werden. Dann muss es selbstverständlich einen Benutzer geben, dessen wichtigste Eigenschaft es wohl ist, welchen Benutzer, Passwort er hat als auch auf welche Seiten er zugreifen darf (und welchen Grupper er angehört). Und, oftmals vergessen, müssen die Zugriffrechte für die entsprechenden Seiten gesetzt sein.

6.5.1 Benutzergruppen

Bevor wir Benutzer anlegen, sollten wir uns überlegen, was ein Benutzer alles machen darf. Hierzu gehört nicht, auf welchen Seiten er was machen darf, sondern was ein Benutzer generell darf und was nicht. In diese Überlegung gehört jeder einzelne Menüpunkt und jede einzelne Funktion innerhalb von Typo3. Dies beinhaltet z.B. auch, welche Optionen für Seiten zur Verfügung stehen sollen. Soll der Benutzer dieser Benutzergruppe eine seiner zu betreuenden Seiten auf einen Zeitraum beschränken, soll er mit Eigenschaften wie „Cache verfällt“ konfrontiert werden oder soll ihm der Button „RichText Editor abschalten“ vorenthalten werden. Sämtliche dieser Eigenschaften können in einer Benutzergruppe angegeben werden. Grundsätzlich ist zunächst alles nicht erlaubt. Erlaubt ist nur, was wir explizit gestatten. Um einen Benutzer entsprechende Rechte geben zu können, ist mindestens eine Benutzergruppe zuvor notwendig, da diese Rechte zum großen Teil nicht in der Benutzer-Konfiguration zur Verfügung stehen.

Benutzergruppen werden als blaues oder rotes Symbol (1 Person) dargestellt. Benutzergruppen (als auch die Benutzer selbst) können nur im root der Baumdarstellung (also der Welt) angelegt werden. Durch einen Klick auf diese Welt können wir eine neue Benutzergruppe anlegen (Backend Benutzergruppe). Auf der nun folgenden Seite können wir die grundlegenden Rechte für diese neue Benutzergruppe angeben.



Hierfür stehen uns diverse unterschiedliche Möglichkeit zur Verfügung. Wichtig ist, das „Include Access Lists“ aktiviert ist.

Die Module könnten später auch bei den einzelnen Benutzern definiert werden. Benutzergruppen haben aber die Eigenschaft, das spätere Änderungen global auf alle Benutzer übertragen werden können.

Die folgenden Eigenschaften hingegen können später nicht mehr bei den Benutzern definiert werden. Sie geben Auskunft darüber, was ein Benutzer später alles darf und was nicht. So stellt sich z.B. die Frage, ob ein Benutzer eine bestehende Seite verändern darf (natürlich nur die Seiten, auf die er später auch Zugriff hat), ob er die Möglichkeit haben soll oder für eine Seite TSConfig-Eigenschaften angeben darf.

Einmal ordnungsgemäß und richtig definiert kann diese Benutzergruppe dann später bei allen Benutzern eingesetzt werden.

6.5.2 Benutzer anlegen (User)

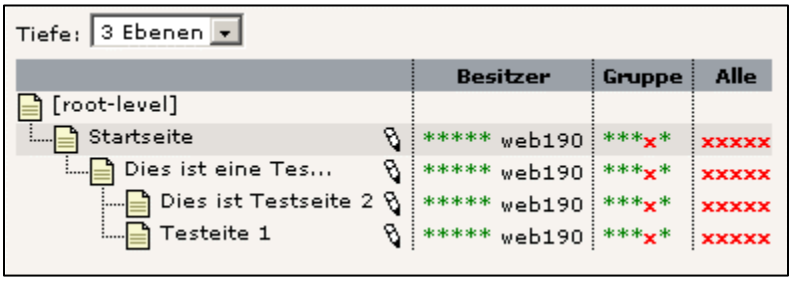
Benutzer werden als blaues oder rotes Symbol (1 Person) dargestellt. Die „roten“ Benutzer sind die Admin-User. Benutzer (als auch Benutzergruppen) können nur im root der Baumdarstellung (also der Welt) angelegt werden. Durch einen Klick auf diese Welt können wir einen neuen Benutzer anlegen (Backend Benutzer). Auf der nun folgenden Seite können wir die grundlegenden Rechte für diesen neuen Benutzer angeben.

Selbstverständlich müssen wir für unseren neuen Benutzer einen Benutzernamen und ein Passwort angeben. Beide Felder sind frei definierbar. Ebenfalls können wir dem Benutzer mehrere Benutzergruppen zuordnen. Eine solche Benutzergruppe ist zwingend erforderlich, da in Ihr die Rechte (durch die Access-Liste) angegeben wurden. Informativ sei hier erwähnt, das sich die Rechte der einzelnen Gruppen und die individuellen Rechte des Benutzer addieren.

Das Feld Admin sollte nur dann aktiviert werden, wenn sichergestellt ist, das der Benutzer wirklich alles darf. In diesem Fall greift das Typo3-Interne Rechtekonzept nicht mehr. Die „Default-Language“ ist im Übrigen irrelevant. Der Benutzer muss später diese Einstellung über user->setup selber vornehmen. Aus diesem Grunde schon sollte die Modul-Liste neben den web-Objekten ebenfalls die user-Objekte beinhalten. Wichtig ist hier zu wissen, das das Grundobjekte „web“ z.B. erforderlich ist, damit z.B. „web_layout“ ebenfalls dargestellt werden kann. Gleiches gilt also auch für „user_setup“. Wichtige Voraussetzung hierfür ist, das ebenfalls „user“ mit in die Liste der ausgewählten Module mit aufgenommen wurde. Als „DB Mounts“ sollten die Übergeordneten Seiten angegeben werden, auf die der Benutzer Zugriff haben darf.

6.5.3 Zugriffsrechte setzen

Nachdem wir nun eine Benutzergruppe als auch einen Benutzer angelegt haben, müssen wir allerdings noch für die entsprechenden Seiten Zugriffsrechte setzen (was leider oftmals vergessen wird). Hierzu gehen wir auf den Punkt „Zugriff“ und erhalten so die Zugriffsrechte für die aktuelle Seite. Durch einen Klick auf das Bleistift-Symbol können wir die Zugriffsrechte einer Seite (oder aber Rekursiv für die Unterseiten) verändern und z.B. auf unsere angelegte Gruppe setzen.



Ab jetzt sollte sich der neue Benutzer erfolgreich mit den gewünschten Möglichkeiten einloggen können.

6.6 Statistiken mit AWStats

Typo3-Seiten werden aus Datenbanken generiert. Hierdurch ergeben sich Dateinamen, die für jedes normale Statistik-Programm lediglich Nummern darstellen würden. Später wird aber noch gezeigt, wie man auch dieses Problem beheben kann. Wo macht es Sinn, das eine Statistik aussagt, das die Seite index.php?ID=2 insgesamt 267 mal aufgerufen wurde, die Seite index.php?ID=419 hingegen nur 59 mal? Auch wenn typo3 Eigenschaften mitbringt, die statische Internet-Seiten simulieren (123.0.html), so sind diese Statistiken dennoch nicht besonders aussagekräftig (Später hierzu aber mehr).

Hierfür bietet TypoScript einige Möglichkeiten, wie ein eigenes Logfile geschrieben werden kann, welches innerhalb der Statistik Seitennamen verwendet, die aussagekräftig sind. Wenige Content Management-Systeme bieten eine solche Statistik-Funktion, die von nahezu jedem Logfile-Analyser gelesen werden kann. AWStats ist hier nur Eines von vielen. Jedoch wurde AWStats so aufbereitet, das es sich direkt in die Typo3-Umgebung integrieren lässt.

Damit man in den Genuss solcher Statistiken kommt, ist es erforderlich, mit Typo3Script einige Werte zu setzen. Wir gehen davon aus, das AWStats bereits installiert und betriebsbereit ist (als über den Extension-Manager hinzugefügt wurde) und die schon oft erwähnte Datei localconf.php im Verzeichnis /typo3conf/ eine entsprechende Wertzuweisung für das Logfile-Verzeichnis hat (logfile_Dir).

Ein Template, sinnvollerweise das Stammtemplate, wird um folgende Einträge ergänzt:

```
config.stat = 1
config.stat_apache = 1
config.stat_apache_logfile = irgendeinname.log
config.stat_mysql = 0
```

Wer möchte, kann den Wert für stat_mysql auch auf 1 setzen:

```
config.stat_mysql = 1
```

Hierdurch wird eine Datenbank-Statistik in der Tabelle sys_stat weitergeführt. Die hier gespeicherten Daten unterscheiden sich jedoch nicht von dem erzeugten Logfile, so dass dieses „Doppeltgemoppelt“ währe.

Die Bezeichnung stat_apache bzw. stat_apache_logfile mag ggf. irreführend sein. Es sei hier erwähnt, das diese Datei im Format einer Apache Logdatei erstellt wird, dies aber keineswegs die Apache-Logdateien ersetzt. Diese werden nach wie vor geschrieben und liegt bei Kunden von typo3server im Verzeichnis /log/ als access_log vor. Hier werden auch die Typo3-Konfigurations-Aufrufe mit gespeichert, was bei den Typo3-Logs nicht der Fall ist.

Der Dateiname einer Logdatei sollte auf .log enden. Bei Typo3server-Präsenzent ist diese Datei ebenfalls schon vorinstalliert. Hier heißt die Datei logfile.log und ist im Verzeichnis fileadmin/ zu finden. Sollte eine solche Log-Datei nicht vorhanden sein, so sei erwähnt, das für einen funktionierenden Einsatz zumindest eine leere Datei mit 0 Byte Inhalt existieren muss, da Typo3 diese nicht von selbst erzeugt.

Wenn man das erste mal AWStats aus dem Tool-Menü der Typo3-Konfiguration aufruft, so wird man nach Domainnamen gefragt. Die Angabe der Domainnamen hat wahrheitsgemäß zu erfolgen, mehrere Domainnamen sind per Komma zu trennen. Die Angabe muss in der Form www.domainname.de sein, also ohne http..., aber mit www....

AWStats aktualisiert sich nicht von alleine, sondern man muss selber aktiv werden, damit man neue Daten erhält. Dieses aktualisieren ist jedoch eine Leichtigkeit, in dem man einfach auf den Link „Jetzt aktualisieren“ klickt.

Aber AWStats hat auch einige Nachteile: Es wird einem vorgetäuscht, das die Internetseiten den Titel der Seite haben, also z.B. „http://www.meinedomain.de/Dies ist meine Homepage“. Eine solche Seite lässt sich natürlich nicht öffnen. Im Vergleich zu den Nummern, also index.php?id=234 oder aber 234.0.html stellt dieses aber der bessere Kompromiss dar. Ebenfalls sei erwähnt, das keine Dateidownloads etc. mit gerechnet werden. Die Anzahl der Hits stimmt z.B. mit der Anzahl der abgerufenen Seiten überein, was ein Indikator dafür ist, das nicht einmal Grafiken mit berechnet werden. Angaben wie z.B. das Transfervolumen sind somit unbrauchbar.

Kapitel 8: TypoScript Kurzreferenz

8.1 Datentypen

Datentyp	Beispiel	Beschreibung
<tag>	<BODY bgcolor="red">	
align	right	right / left / center Default: left
VHalign	r, c	r/c/l, t/c/b Horizontal (right, center, left) Vertikal (top, center, bottom) 2 Werte, mit Komma separiert Default: l,t
resource	1.) fileadmin/bild.gif 2.) bild*.gif	Es gibt zwei Arten von Ressourcen (Dateien): Eine direkte Dateiangabe incl. Pfad sowie ein Verweis auf eine Datei innerhalb der Datenbank (z.B. Datei innerhalb eines Templates). Befindet sich innerhalb des Wertes ein „/“, wird die Datei aus dem angegebenen Verzeichnis genommen. Ansonsten wird z.B. innerhalb des Templates (Resources) nach einer Datei gesucht. Das Sternchen bewirkt, dass die letzte Dateiversion (z.B. toplogo_05.gif) verwendet wird. Den numerischen Prefix legt Typo3 beim Hochladen selbstständig an, wenn der Dateiname bereits vergeben ist.
imgResource	fileadmin/bild.gif bild*.gif	(Ähnlich wie resource, s.o.) imgResource erwartet als Datei eine gültige Grafik. Welche Dateitypen erlaubt sind, wird in $TYPO3_CONF_VARS[„GFX“] [„imagefile_ext“] festgelegt (Installtool bzw. localconf.php). Default: pdf, gif, jpeg, jpg, tif, bmp, ai, pcx, tga, png
target	_top _blank page	HTML-Target, der in <A>-Tags Verwendung findet. Wird überwiegend bei Frames benötigt. Der Datentyp target verdeutlicht, dass ein HTML-Target erwartet wird. Als Wert wird jede beliebige Zeichenkette akzeptiert.
degree		Mögliche Werte sind -90 bis 90

int+ / posint		Positive Zahl (Integer-Wert größer oder gleich 0)
int		Beliebige Zahl. Verwendung ist oftmals allgemein – int+ ist z.B. oftmals besser angebracht.
string / str / value		Beliebige Zeichenkette. Verwendung ist oftmals allgemein – andere Datentypen (z.B. target) sind angebrachter
boolean	1	True / False, 1/0, ja/nein Leere Zeichenketten werden als „falsch“ gewertet, Beliebige Zeichenkette (außer „=0“) werden als „wahr“ gewertet. Üblicherweise werden Werte wie 1 für „wahr“ und 0 für „falsch“ verwendet.
rotation		Wie int+, jedoch verdeutlicht rotation, dass eine Zahl zwischen 0 und 360 erwartet wird.
x,y,w,h	10,10,5,5	x, y gibt die Position von einer linken oberen Ecke an (x=nach rechts, y=nach unten). w, h gibt eine Breite und eine Höhe an.
HTML-color	red #ffeecc	Übliche HTML-Farbangaben.
GraphicColor	red #ffeecc 255,0,255	Wie HTML-color, jedoch ist zusätzlich die Angabe von RGB-Werten möglich.
page_id	34 this	Angabe einer Seiten-ID. page-id Erwartet einen positive Zahl oder aber die Angabe von „this“ als Verweis auf die aktuelle Seite.
list	item1, item2, item3	Kommaseparierte Liste von beliebigen Zeichenketten
wrap	 | 	Ein Wert kann beliebigen Zeichenketten eingeschlossen werden. Im Beispiel wird dem Wert vorangestellt, wird am Ende hinzugefügt.
case	upper	upper / lower
getText		Erläuterungen siehe Abschnitt „data (getText)“ (Seite **Fehler! Textmarke nicht definiert.**).
dir	fileadmin/files | pdf, gif, jpg | name | r | true	Gibt eine Dateiliste aus. Hier: Aus dem Ordner fileadmin/files werden alle pdf, gif und jpg-Dateien aufgelistet, sortiert nach dem Dateinamen. Das Durchsuchen erfolgt rekursive (inkl. Unterverzeichnisse), dem Dateinamen wird der gesamte Pfad vorangestellt. Syntax: [relativer Pfad] | [Liste von Dateiendungen] | [Sortierung: name, size, ext, date] | [reverse: „r“] | [vollständige Pfadangabe: Boolean]

8.2 Objektgruppen

An diversen Stellen in der Referenz wird auf eine Objektgruppe verwiesen. Welche Objekte zu einer Objektgruppe gehören, können Sie der folgenden Liste entnehmen.

Objektgruppe	Einzelne, untergeordnete Objekte
cObjekt	HTML, TEXT, IMAGE, …
frameObj	FRAMESET, FRAME
menuObj	GMENU, TMENU, IMGMENU, JSMENU
GifBuilderObj	TEXT, SHADOW, OUTLINE, EMBOSS, BOX, IMAGE, EFFECT

8.3 Funktionen / stdWrap

Eigenschaft	Datentyp	Beschreibung
8.3.1 Daten auslesen		
field	fieldname	Gibt den Inhalt eines angegebenen Feldnamens aus. Beispiel: seite.10 = TEXT seite.10.field = title (Dies gibt den Titel der aktuellen Seite aus). field ist die einfache Form von getData. Hinweis: Mit „.field = nav_title // title“ wird der Inhalt von title nur dann ausgegeben, wenn nav_title keinen Wert enthält.
data	getText	Nähere Informationen im Abschnitt „data (getText)“ (Seite **Fehler! Textmarke nicht definiert.**)
cObject	cObject	Der Inhalt eines anderen cObjektes (z.B. TEXT, IMAGE etc.) wird ausgegeben.
filelist	dir / stdWrap	Siehe „Datentypen -> dir“

8.3.2 data (getText)	
[.data = …]	
Mit der Funktion data können diverse Werte ausgelesen werden.	
field : [Feldname]	Mit field kann ein beliebiges Feld aus der Tabelle pages des aktuellen Seite ausgelesen werden. Indentisch zur stdWrap-Funktion field. Beispiel: 10 = TEXT 10.data = field : header 20 = TEXT 20.field = header
getenv : HTTP_REFERER	Mit genenv können die üblichen Umgebungsvariablen, wie z.B. der Referer ausgelesen werden. Beispiel: 10 = TEXT 10.data = getenv : HTTP_REFERER
date : dd-mm-yy	Mit date kann das aktuelle Datum zurückgeliefert werden. Beispiel: 10 = TEXT 10.data = date : d-m-y
GPvar : irgendwas	Mit GPvar können GET- und POST-Werte ausgelesen werden (z.B. bei Formularen etc.). Beispiel: 10 = TEXT 10.data = GPvar : username
TSFE : [TSFE-Wert]	Mit TSFE können Werte ausgelesen werden, die im globalem TSFE-Array stehen. Beispiel: 10 = TEXT 10.data = TSFE : loginUser
DB : tt_content : 12 : header	Mit DB kann ein beliebiger Datensatz ausgelesen werden. Syntax: DB : [Tabelle] : [uid] : [Feld] Hinweis: Es können hier nur Felder angezeigt werden, die in TCA enthalten (z.B. im Backend sichtbar). Ebenfalls werden Datensätze, die als deleted gekennzeichnet sind, nicht ausgegeben.
Hinweis: Mit // können mehrere Werte angegeben werden. Es wird der erste Wert genommen, der ein Ergebnis größer als 0 enthält. Beispiel: 10.data = GPvar : name // GPvar : username	

Eigenschaft	Datentyp	Beschreibung
8.3.3 Bedingungen		
override	string / stdWrap	Wenn “override” einen Wert größer als „“ oder 0 zurückliefert, wird der Wert genommen, der sich hinter override ergibt.
ifEmpty	string / stdWrap	Wenn der Inhalt bis zu diesem Punkte leer ist (oder = 0), dann wird der Wert genommen, der sich hinter ifEmpty ergibt.
required	boolean	Wird required auf 1 gesetzt, wird das gesamte Objekt nur dann ausgeführt, wenn tatsächlich ein Wert zurückgegeben wurde. Beispiel: Dieses Beispiel führt jeden Datensatz nur dann aus, wenn im Datenbankfeld bodytext auch ein Wert enthalten ist. seite.10 = TEXT seite.10.field = bodytext seite.10.wrap = <hr size=“1“>| seite.10.required = 1
fieldRequired		Ähnlich wie required, jedoch kann hier speziell auf ein bestimmtes Feld hin geprüft werden. Beispiel: Dieses Beispiel führt jeden Datensatz nur dann aus, wenn im Datenbankfeld title auch ein Wert enthalten ist. seite.10 = TEXT seite.10.field = bodytext seite.10.fieldRequired = title
if		Nähere Informationen im Abschnitt “if” (Seite 184).

Eigenschaft	Datentyp	Beschreibung
8.3.4 Parse-Funktionen		

listNum	int
	+calc
	+“last“
		Teilt einen Inhalt nach Kommata (oder einem anderen Zeichen) auf (explode).
		
		Besondere (weitere) Eigenschaften:
		splitChar: Trennzeichen, Default = Komma. Wird ein numerischer Wert angegeben (z.B. „10“), wird das entsprechende Ascii-Zeichen genommen (10=Zeilenumbruch)
		
		Beispiel: Dieses Beispiel zeigt aus dem Datenbankfeld „bodytext“ nur den letzten Satz an.
		seite.10 = TEXT
		seite.10.field = bodytext
		# Char 46 ist ein Punkt
		seite.10.splitChar = 46
		seite.10.listNum = last – 1

parseFunc		Nähere Informationen im Abschnitt „parseFunc“ (Seite 186).
split		Nähere Informationen im Abschnitt „split“ (Seite 183).
trim		Entfernt Leerzeichen vor und hinter einem Wert.
intval	boolean	Wandelt einen Wert in eine Zahl um.
case	case	Konvertiert eine Zeichenkette in Groß- oder Kleinschreibung um. Mögliche Werte sind upper und lower.
crop		Schneidet den Inhalt nach einer bestimmten Anzahl von Zeichen ab und fügt beliebige Zeichen an. Beispiel: Dieses Beispiel liefert als Ergebnis „Dies ist nu...“ zurück. Der Rest wird abgeschnitten und durch die angegebenen Punkte ersetzt. seite.10 = TEXT seite.10.value = Dies ist nur ein Test seite.10.crop = 11 | ...
stripHtml	boolean	Entfernt alle HTML-Tags aus dem Inhalt
htmlSpecialChars	boolean	Der Inhalt wird durch die PHP-Funktion htmlspecialchars() geparst. Die Auswirkungen sind, das HTML-Code nicht ausgeführt, sondern direkt angezeigt wird.

br	boolean	Konvertiert alle Zeilenumbrüche in einen -Tag (bzw. wie unter brTag angegeben).
brTag	string	Alle ASCII-Codes mit “10” (CR) werden durch dem angegebenen Wert ersetzt. Beispiel: brTag =
doubleBrTag	string	Alle doppelten Line-Breaks (ASCII-Code „10“) werden mit dem angegebenen Wert ersetzt. Beispiel: doubleBrTag = <p>

Eigenschaft	Datentyp	Beschreibung
8.3.5 Datums- und Zeitfunktionen		
date	date-conf	Formatiert einen Wert in ein Datum
strftime	strftime-conf	Wie date (s.o.)
age	Boolean String: Min | Std | Tage | Jahre	Wenn age = 1 gesetzt wird, wird der Inhalt (z.B. zurückgelieferter Wert aus einer Datenbankabfrage) als ein Datum gesehen und die Differenz zwischen dem aktuellen Datum und dem zurückgeliefertem Datum berechnet. Beinhaltet age eine Zeichenkette, so dient diese Zeichenkette zur Formatierung (4 Werte für Minuten, Stunden, Tage und Jahre jeweils getrennt mit einem | (Pipe)-Symbol.

Eigenschaft	Datentyp	Beschreibung
8.3.6 EditPanel		
editPanel	boolean / editPanel	Siehe Objekt “EDITPANEL”
editIcons	string	Siehe Objekt “EDITPANEL”

Eigenschaft	Datentyp	Beschreibung
8.3.7 Debugging		
debug	boolean	Ähnlich wie htmlSpecialChars: Liefert das Ergebnis in HTML-Darstellung zurück. HTML-Code wird somit nicht vom Browser ausgefürt.
debugFunc	Boolean / Integer 2 =Darstellung in einer Tabelle	Liefert den Inhalt zurück. mit debugFunc = 2 können einzelne Werte in einer Tabelle überprüft werden.
debugData	boolean	Liefert den Inhalt von $cObj->data direkt an den Browser aus.

Eigenschaft	Datentyp	Beschreibung
8.3.8 imgResource		
ext	imageExtension / stdWrap	Gibt an, welche Dateiendungen als imgResource gewertet werden sollen. Default ist „web“ (=gif, jpg)
width height	int / stdWrap	Gibt die gewünschte Breite und die Höhe einer Grafik in Pixel an. Hinweis: Wenn beide Parameter angegeben wurden und einem der Parameter wurde hinter der Pixelangabe ein „m“ angestellt, bleiben die Proportionen erhalten und die angegebenen Größen gelten als maximale Dimensionen.
params	string	Beliebige Zeichenkette. Hier können an ImageMagick beliebige Parameter übergeben werden (per Kommandozeile). Beispiel: -rotate 90 Nähere Informationen finden Sie in der Dokumentation zu ImageMagick
frame	int	Gibt bei animierten Gifs und bei PDF-Dateien den gewünschten Frame an.
import	path / stdWrap	Mit import wird ein Verzeichnis bestimmt, aus dem eine Grafik geladen werden soll. Beispiel: .import = fileadmin/pics/ .import.field = image .import.listNum = 0
maxW maxH	int / stdWrap	maximale Breite (maxW) und maximale Höhe (maxH) in Pixel
minW minH	int / stdWrap	minimale Breite (minW) und minimale Höhe (minH) in Pixel

Eigenschaft	Datentyp	Beschreibung
8.3.9 imageLinkWrap		
enable	boolean / stdWrap	Aktiviert das Erzeugen eines Links zu einer Grafiken.
width height	int (1-1000)	Breite und Höhe der Grafik in Pixel. Durch ein „m“ hinter einem der Werte (z.B. .width=100m) werden die Proportionen beibehalten – width und height sind dann maximale Werte.
effects	string	Hier können besondere Effekte angegeben werden, z.B. .effects = gamma = 1,3 | sharpen=80 Nähere Informationen unter GIFBUILDER -> effects (Seite
title	string	Seitentitel des neuen Fensters (HTML)
bodyTag	<tag>	Body tag des neuen Fensters Beispiel: <body bgcolor = “white”>
wrap	wrap	Wrap um das Bild. Beispiel: <table border=“1“><tr><td> | </td></tr></table>
JSwindow	boolean	Das Bild wird in einem neuen Fenster geöffnet, die Dimensionen des neuen Fensters passen sich der Bildgröße an.
JSwindow.expand	x, y	Gibt an, um welche Pixelzahl das Fenster größer sein soll als das Bild.
JSwindow.newWindow	boolean	Wenn gesetzt, wird jedes Bild in einem neuen Fenster geöffnet.

Eigenschaft	Datentyp	Beschreibung
8.3.10 numRows		
table	string	Angabe des Tabellennamens.
select	-> select	Nähere Informationen im Abschnitt “select” (Seite **Fehler! Textmarke nicht definiert.**).

Eigenschaft	Datentyp	Beschreibung
8.3.11 select		
uidInList		Kommaseparierte Liste von Unique-IDs
pidInList		Kommaseparierte Liste von Parent-IDs
orderBy	SQL: Order By	Beispiel: select.orderBy = sorting, title
groupBy	SQL: Group By	Beispiel: select.groupBy = CType
max	int +calc +”total”	Es werden maximal x Datensätze ausgegeben. „total“ = count(*)
begin	int +calc +“total“	Es wird erst ab dem x-ten Datensatz begonnen. „total“ = count(*)
where	SQL: Where	Erweiterung der Where-Klausel um eigene Bedingungen. Beispiel: select.where = colPos = 0 AND CType=’text’

Eigenschaft	Datentyp	Beschreibung
8.3.12 split		
token	string / stdWrap	Trennzeichen Auch: token.char = 10 (Zeilenumbruch als Trenner)
max	int / stdWrap	Maximale Anzahl an Aufteilungen
min	int / stdWrap	Minimale Anzahl an Aufteilungen
cObjNum	cObjNum + optionSplit	
1,2,3,4	-> CARRAY / stdWrap	Das Objekt, das den Wert bearbeiten soll. Beispiel: 1.current = 1 1.wrap = |
wrap	wrap + optionSplit	Gibt einen Wrap für jeden einzelnen Eintrag an.

Eigenschaft	Datentyp	Beschreibung
8.3.13 if		
Die „if-Funktion“ liefert „wahr“ zurück, wenn alle(!) verwendete Abfragen „wahr“ zurückliefern. Wenn eine einzelne Abfrage „falsch“ zurückliefert, ist das Gesamtergebnis falsch.		
isTrue	string / stdWrap	Wenn der Inhalt „wahr“ ist (kein leerer String und ungleich 0)...
isFalse	string / stdWrap	Wenn der Inhalt „falsch“ ist (leerer String oder String = 0)...
isPositive	int / stdWrap + calc	Liefert „wahr“ zurück, wenn der Inhalt eine positive Zahl ist
isGreaterThan	.value / stdWrap	Liefert „wahr“ zurück, wenn der Inhalt größer ist als der Wert von „isGreaterThan.value“.
isLessThan	.value / stdWrap	Liefert “wahr” zurück, wenn der Inhalt kleiner ist als der Wert von “isLessThan.value“.
equals	.value / stdWrap	Liefert “wahr” zurück, wenn der Inhalt identisch ist mit dem Wert von “equals.value”.
isInList	.value / stdWrap	Liefert “wahr” zurück, wenn der Inhalt in der kommaseparierten Liste von „isInList.value“ vorhanden ist. Hinweis: Die kommaseparierte Liste darf zwischen den einzelnen Elementen keine Leerzeichen haben!
value	string / stdWrap	Ggf. Angabe eines zu überprüfenden Wertes.
negate	boolean	Das zurückgelieferte Ergebnis (Wahr / Falsch) wird negiert.

Eigenschaft	Datentyp	Beschreibung
8.3.14 encapsLines		
encapsTagList	Kommaseparierte Stringliste	Liste von HTML-Tags, die überprüft (und ggf. ersetzt) werden soll. Einzelne Werte müssen „lowercase“ angegeben werden. Beispiel: encapsTagList = div, p
remapTag.[tag]	string	Einen verkapselten (<tag>x</tag>) HTML-Tag durch einen anderen Ersetzen. Beispiel: remapTag.P = DIV Jeder <P>-HTML-Tag wird in einen <DIV>-Tag umgewandelt. Hinweis: Der [tag] muss in uppercase angegeben werden!

addAttributes.[tag]	strings (array)	Fügt jedem dieser Tags angegebene Parameter hinzu.
		Beispiel:
		addAttributes.P {
		style = margin-bottom:1px
		align = right
		}
		Hinweis:
		Der [tag] muss in uppercase angegeben werden!

Eigenschaft	Datentyp	Beschreibung
8.3.15 parseFunc		
short	Strings (Array)	Hier können Abkürzungen vereinbart werden. Beispiel: .short.t3 = Typo3 .short.versionx = 3.6.2 .short.backToHome = <a href="index.php"> Alle Vorkommnisse von "t3" werden in „Typo3" umgewandelt. Beim zweiten Beispiel wird ein Anwendungsfall demonstriert, wie z.B. sich eine ändernde Versionsnummer innerhalb der Contents dynamisch gehalten werden kann. Das dritte Beispiel demonstriert das Setzen eines Links auf die Homepage.
makelinks	boolean / stdWrap	makelinks = 1 erzeugt aus jedem Content, der mit "http://" oder "mailto:" beginnt, einen Link.

tags	String (Array)	Eigene HTML-Tags
		Beispiel:
		.tags.fett = TEXT
		.tags.fett {
		current = 1
		wrap = |
		}

denyTags	Strings (Kommasepariert)	In „denyTags" können HTML-Tags verboten werden. Beispiel: .denyTags = font, span
allowTags	Strings	Hier werden erlaubte Tags aufgeführt. Wird ein Tag in „allowTags" gefunden, wird „denyTags" ignoriert.

8.4 Objekt-Referenz

8.4.1 PAGE		
Eigenschaft	Typ	Beschreibung
typeNum	Integer	Angabe der typeNum, die das gewünschte „PAGE“-Objekt auswählt. Hinweis: typeNum muss gesetzt werden und eindeutig sein.
1,2,3,10,20,30	cObject	
wrap	wrap	Inhalt wird gewrapt
stdWrap	-> stdWrap	Wrapt den Inhalt mit den stdWrap-Eigenschaften und Funktionen.
bodyTag	<tag>	Body-Tag (HTML) der Seite Beispiel: seite = PAGE seite.typeNum = 0 seite.bodyTag = <body bgcolor = “red”> Default: <body bgcolor = ”#FFFFFF”>
frameSet	Objekt FRAMESET	Nähere Informationen im Abschnitt „Frameset“
meta	Objekt META	Angabe von Meta-Tags Beispiel: seite = PAGE seite.typeNum = 0 seite.meta.AUTHOR = R. Meyer seite.meta.KEYWORDS.field = title
shortcutIcon	resource (.ico)	Der MSIE kann kleine Icons zur URL hinzufügen, wenn diese im Bookmark enthalten ist. Hier wird ein Verweis zu einer .ico-Datei angegeben.
headerData	string	Fügt beliebigen Code (z.B. JavaScript-Code etc.) in den Header-Bereich ein.
config	-> CONFIG	Nähere Informationen ab Abschnitt „CONFIG“
includeLibs		
stylesheet	resource	Fügt in den Header-Bereich der Seite eine css-Datei ein. Beispielresultat: <link rel=“stylesheet“ href=”fileadmin/meinedatei.css”>

includeCSS.[array]	resource (Array)	Wie Eigenschaft “stylesheet”, jedoch können hier mehrere Stylesheets angegeben werden. Beispiel: seite = PAGE seite.typeNum = 0 seite.includeCSS.f1 = fileadmin/f1.css seite.includeCSS.f2 = fileadmin/f2.css
CSS_inlineStyle	string	Hier kann beliebiger CSS-Code direkt an den Browser „durchgereicht“ werden. (Inline-CSS bzw. in-dokument CSS). Der <style>-Tag selbst wird von Typo3 erzeugt.

8.4.2 TEXT		
Eigenschaft	Typ	Beschreibung
value	string	Ausgabe eines beliebigen Textes. Beispiel: seite.10 = TEXT seite.10.value = Hallo Welt
(stdWrap-Eigenschaften)		Beispiel: seite.10 = TEXT seite.10.field = title

8.4.3 COA		
Eigenschaft	Typ	Beschreibung
1,2,3,4,10,20	cObject	Mit COA kann aufgesplittet werden Beispiel: seite = PAGE seite.typeNum = 0 seite.10 = COA seite.10.10 = TEXT seite.10.10.value = Hallo seite.10.20 = TEXT seite.10.20.value = Welt
if	-> if	Wenn „if“ „falsch“ zurückliefert, wird der COA-Abschnitt nicht gerendert.
wrap	wrap	
stdWrap	stdWrap	

8.4.4 CASE		
Eigenschaft	Typ	Beschreibung
key	string / stdWrap	In Key wird der zu überprüfende Wert eingetragen. Beispiel: .10 = CASE .10.key.field = layout
[value-array]	cObject	Hier können die beliebigen Werte angegeben werden, nach denen eine CASE-Abfrage überprüft werden soll. Beispiel: .10 = CASE .10.key.field = layout .10.**1** = TEMPLATE […] .10.**7** = TEXT […] Die 1 als auch die 7 spiegeln in diesem Beispiel gespeicherte Werte des Datenbankfeldes „layout“ wieder: Steht in dem Feld „Layout“ eine 1, wird das TEMPLATE-Objekt ausgeführt, steht in dem Feld „Layout“ eine 7, wird das TEXT-Objekt ausgeführt.
default	cObject	Default, wenn keine Übereinstimmung gefunden wurde.
stdWrap	-> stdWrap	
if	-> if	Nur wenn “if” einen wahren Wert (>0) zurückliefert, wird die CASE-Abfrage ausgeführt.

8.4.5 FILE		
Eigenschaft	Typ	Beschreibung
file	resource	Die unter .file angegebene Datei wird direkt an den HTML-Code übergeben. Ausnahmen sind jpg, jpeg, gif und png-Dateien: Diese werden direkt in einen img-Tag umgewandelt. Beispiel: .10 = FILE .10.file = fileadmin/datei.txt Hinweis: Die Dateigröße ist intern auf 1 MByte beschränkt.
wrap	wrap	

8.4.6 TEMPLATE		
Eigenschaft	Typ	Beschreibung
template	cObject	Unter der Eigenschaft „template“ wird ein Objekt angegeben, in dem die Designvorlage abgelegt ist. Sinnvolle Objekte sind hier TEXT (mühselig) und FILE (ausgelagert). Beispiel: .10 = TEMPLATE .10.template = FILE .10.template.file = fileadmin/datei.html
marks.[Platzhalter]	cObject	Unterhalb von marks werden Platzhalter definiert. Dabei wird direkt hinter marks. der Platzhalterbezeichner angegeben und dieses einem Objekt zugewiesen. Beispiel: .10 = TEMPLATE .10.template = FILE .10.template.file = fileadmin/datei.html .10.marks.INHALT = TEXT .10.marks.INHALT.value = Hallo Welt Hinweis: Die Platzhalterbezeichner sind case-sensitive!
workOnSubpart	string	In dem oben genannten Beispiel (von marks) wird eine gesamte Datei eingelesen. Mit „WorkOnSubpart“ kann auf einzelne Teilbereiche einer Datei zurückgegriffen werden, wenn dieses angegeben sind. Solche Teilbereiche werden in der Regel mit <!-- ###SUBPARTBEZEICHNER### BEGIN--> und <!-- ###SUBPARTBEZEICHNER### END--> eingeschlossen. Im Wesentlichen besteht ein Subpart aus zwei Markierungen, ähnlich einem Platzhalter, die einen Teilbereich einschließen.

8.4.7 CONTENT		
Eigenschaft	Typ	Beschreibung
table	string	Hier wird der Name einer Datenbanktabelle angegeben. Hinweis: Mögliche Tabellen sind „pages“ sowie alle Tabellen mit dem Prefix „tx_“, „tt_“, „ttx_“, „fe_“, „user_“. Beispiel: .10 = CONTENT .10.table = tt_content
select	-> select	Hier kann in die SQL-Abfrage eingegriffen werden. Beispiel: .10 = CONTENT .10.table = tt_content .10.select.orderBy = sorting Nähere Informationen im Abschnitt „select“ (Seite **Fehler! Textmarke nicht definiert.**).
wrap	wrap	
stdWrap	stdWrap	

8.4.8 IMAGE		
Eigenschaft	Typ	Beschreibung
file	imgResource	
params	string	Angabe von Parameter für den img-Tag
alttext	string / stdWrap	Beispiel: .10 = IMAGE .10.file = fileadmin/firmengebaeude.gif .10.alttext = Unsere Firma
if	-> if	Nur wenn „if” einen wahren Wert zurückliefert, wird IMAGE ausgeführt.
wrap	wrap	
stdWrap	-> stdWrap	

8.4.9 GIFBUILDER		
Eigenschaft	Typ	Beschreibung
XY	x, y (Kalkulation möglich)	Mit XY werden die gesamten Dimensionen der zu erzeugenden Grafik angegeben.
format	gif / jpg	Durch Angabe eines Formates kann manuell auf die Dateigröße zugegriffen werden. Standard ist immer gif. Beispiel: seite.10 = IMAGE seite.10.file = GIFBUILDER seite.10.file.format = jpg
reduceColors	Integer (1-255)	Sofern das format ein gif-File ist (Standard), können die Farben reduziert werden. Mögliche Werte sind 1 bis 255. Beispiel: seite.10 = IMAGE seite.10.file = GIFBUILDER seite.10.file.reduceColors = 16
quality	Integer (10-100)	Wenn das Format der Grafik jpg ist, dann kann mit quality die Qualität der jpg-Grafik angegeben werden.
transparentBackground	Boolean 1/0	Wenn diese Eigenschaft auf 1 gesetzt wird, wird die Grafik mit der Farbe transparent gemacht, die auf der Position 0.0 (Pixel linke obere Ecke) der Grafik gefunden wird. Hinweis: niceText sollte mit dieser Eigenschaft nicht zusammen verwendet werden.
transparentColor	HTML-Color	Hier kann explizit eine transparente Farbe angeben werden. Die Beschreibung kann dabei in 3 Arten erfolgen: #ffffcc red 255,255,127 Hinweis: niceText und reduceColors sollten mit dieser Eigenschaft nicht zusammen verwendet werden. Beispiel: seite.10 = IMAGE seite.10.file = GIFBUILDER seite.10.file.transparentColor = #f3c6cc
transparentColor.closest	boolean 1/0	Transparente Farbe wird die nächstliegende Farbe.
backColor	Color	Hintergrundfarbe für die gesamte Grafik. Standard ist ein weißer Hintergrund. Beispiel: seite.10 = IMAGE

		seite.10.file = GIFBUILDER seite.10.file.backColor = #333333
maxWidth	Integer / Pixel	Mit maxWidth kann die maximale Breite der Grafik angegeben werden. Diese Eigenschaft kommt insbesondere bei dynamischer Angabe von XY zum Einsatz. Beispiel: seite.10 = IMAGE seite.10.file = GIFBUILDER seite.10.file.XY = [10.w], [10.h]+20 seite.10.file.maxWidth = 160
maxHeight	Integer / Pixel	Siehe maxWidth

8.4.10 HMENU		
Eigenschaft	Typ	Beschreibung

1, 2, 3 etc.	Integer	Gibt an, auf welcher Menüebene gearbeitet werden soll. 1 ist z.B. die oberste Ebene, wohingegen 2 schon die ersten Unterordner sind, die andere Eigenschaften erhalten können.
		Beispiel:
		seite.10.marks.MENU_LINKS = HMENU
		seite.10.marks.MENU_LINKS {
		1 = TMENU
		1.10 = [...]
		2 = TMENU
		2.10 = [...]
		}

entryLevel	Integer	Der Einstiegslevel für dieses Menü. Gibt an, ab welcher Ebene ausgehend von der rootline (-1) das Menü angezeigt werden soll.
special	String	Mit special können die Menüelemente selektiert werden. Diese Eigenschaft ist nicht zwingend erforderlich, für manchen Realisierungswünsche ist sie jedoch sehr nützlich. Mögliche Werte: directory, list, updated, browse, rootline, keyword **Hinweis:** special kann weitere Untereigenschaften wie z.B. special (bei directory) oder range (bei rootline) haben.
minItems	Integer	Die Mindestanzahl von Menüeinträgen. Wenn nicht genügend Menüeinträge vorhanden sind,

		werden Dummy-Einträge mit dem Titel ". . ." mit einem Link auf die aktuelle Seite erzeugt.
maxItems	Integer	Die maximale Anzahl von Menüeinträgen. Falls mehr Menüeinträge vorhanden sind, werden diese bei der Menüerzeugung ignoriert.
excludeUidList	Integer (komma-separierte Liste)	Hier können Seiten (uid's) angegeben werden, die nicht im Menü erscheinen sollen. Hierdurch lassen sich bestimmte Seiten einfach verstecken.
begin	Integer	Der erste Eintrag im Menü. Beispiel: Um die ersten zwei Elemente eines Menüs nicht anzuzeigen, kann man hier begin = 3 angeben.
wrap	wrap	
stdWrap	stdWrap	

8.4.11 TMENU / TMENU_ITEM		
Eigenschaft	Typ	Beschreibung
expAll	Boolean	Wenn expAll auf 1 gesetzt ist, werden die Untermenüs ebenfalls mit angezeigt (so wie z.B. im Beispiel 2). Wenn dies auf 0 gesetzt ist, werden jeweils nur die Untermenüs des aktuellen Eintrages angezeigt und „öffnet" somit eine Art Ordner.
allWrap	wrap	wrapt das gesamte Menüelement mit HTML-Code. Beispiel: seite.10 = GMENU seite.10.allWrap = |
linkWrap	wrap	

8.4.12 GMENU / GMENU_ITEM		
Eigenschaft	Typ	Beschreibung
expAll	Boolean	Wenn expAll auf 1 gesetzt ist, werden die Untermenüs ebenfalls mit angezeigt (so wie z.B. im Beispiel 2). Wenn dies auf 0 gesetzt ist, werden jeweils nur die Untermenüs des aktuellen Eintrages angezeigt und somit eine Art Ortner geöffnet.
collapse	Boolean	Gibt an, was beim Verhalten eines „geöffneten“ Elementes passieren soll. Wenn collapse = 1 ist, wird durch ein Klick auf ein bereits geöffnetes Menüelement dieser „Ordner“ wieder geschlossen. Die Untermenüs dieses Eintrages werden somit nicht mehr angezeigt. Dieses macht nur in Kombination mit expAll=0 Sinn.
target	string (HTML-target)	Hier kann angegeben werden, was der Target eines Links sein soll. Übliches HTML. Beispiel: marks.MENU_OBEN.1.target = _self
wrap	wrap	Wenn es Menüelemente gibt, kann hier der wrap um das „gesamte Menüsgebilde“ angegeben werden.
forceTypeValue	Integer	Angabe eines expliziten &type-Parameters, der für den Link verwendet werden soll.
min	x, y (Kalkulation möglich)	Mit min können die minimalen Dimensionen des gesamten Menüs angegeben werden. Beispiel: marks.MENU_OBEN.1.min = 100,20 Hierdurch ist das gesamte Menü mindestens 100 Pixel breit und 20 Pixel hoch.
max	x, y (Kalkulation möglich)	Wie min, jedoch werden hier die maximalen Dimensionen des gesamten Menüs angegeben.
useLargestItemX	Boolean	Die gesamte Breite des Menüs ist so breit, wie das längste Element.
useLargestItemY	Boolean	Die gesamte Höhe des Menüs ist so hoch, wie das höchste Element.

disableAltText	Boolean	Wenn diese Eigenschaft auf 1 gesetzt wird, werden keine Alt-Tags mehr erzeugt.
before	String / HTML	Gibt an, was vor einem Menüeintrag angezeigt werden soll.
beforeImg	String / Dateipfad	Absolute Angabe einer Grafikdatei, die vor dem Menüeintrag angezeigt wird. Beispiel: seite.10.1.beforeImg = /fileadmin/1a.gif

8.4.13 EDITPANEL		
Eigenschaft	Typ	Beschreibung
label	String	Titel für das Editpanel. Der aktuelle Datensatz-Titel kann mit %s eingefügt werden. Beispiel: seite.20 = EDITPANEL seite.20.label = Sie editieren: %s
allow	string (Komma-separiert)	Gibt an, welche Funktionen zur Verfügung stehen. Folgende Optionen können angegeben werden: toolbar, edit, new, delete, move, hide Beispiel: seite.20 = EDITPANEL seite.20.allow = edit, new, delete
newRecordFromTable	string / Name der Tabelle	Gibt an, für welche Tabelle neue Datensätze angelegt werden. Beispiel: seite.20 = EDITPANEL seite.20. newRecordFromTable = pages
line	integer	Hier kann die Distanz einer Linie (1 Pixel) zum Editpanel angegeben werden. line=0 zeigt keine Linie an.
edit.displayRecord	boolean	Wenn diese Eigenschaft auf 1 gesetzt wird, wird der editierte Eintrag oberhalb des Editierformulares angezeigt.
onlyCurrentPid	boolean	Wenn diese Eigenschaft auf 1 gesetzt wird, werden nur Datensätze mit einem Editpanel versehen, wenn diese Datensätze tatsächlich auf der aktuellen Seite liegen. Mit der CONTENT->select Eigenschaft können z.B. Inhalte fremder Seiten eingelesen werden.

8.4.14 FORM		
Eigenschaft	Typ	Beschreibung
layout		Hier wird das Layout festgelegt. Dabei wird mit Markern gearbeitet. Mögliche Marker sind ###label### und ###field###, was jeweils den Text vor einem Formularfeld angibt und das Formularfeld selbst. Beispiel: seite.10.marks.SUCHE.layout = <tr><td>###LABEL###</td><td> ###FIELD###</td>
wrap		Hier wird angegeben, was um das gesamte Formular gewrappt wird. Beispiel: seite.10.marks.SUCHE.wrap = <table>|</table>
target		Hier wird der Target des form-Tags angegeben. Beispiel: seite.10.marks.SUCHE.target = _self
badMess	string	Wenn NICHT alle Felder ausgefüllt wurden, wird dieser Hinweistext ausgegeben.
goodMess	string	Hier wird angegeben, welcher Hinweistext ausgegeben werden soll, wenn alle Felder ausgefüllt wurden. Besser die Eigenschaft redirect (s.u.) verwenden.
redirect	integer	Hier wird angegeben, auf welche Seite (uid) verwiesen werden soll, wenn das Formular korrekt ausgefüllt wurde. Beispiel: seite.10.marks.SUCHE.redirect = 123
recipient	string	eMail-Adresse des Empfängers, wenn bei der Konfiguration formtype_mail=submit angegeben wurde.
data		Hier stehen die einzelnen Formularfelder. Wird oft in Kombination mit „field“ verwendet, z.B. data.field = Datenbankfeld.
dataArray		Wie data, jedoch können hier mehrere Objekte aufgenommen werden (z.B. für Auswahl-Felder),
image	string / Objekt	Angabe einer Grafik für den Submit-Button. Beispiel: 30.image = FILE 30.image.file = fileadmin/submit.gif

8.4.15 USER / USER_INT

Der Unterschied zwischen USER und USER_INT liegt insbesondere an der Handhabung des Caches. Das Resultat aus USER_INT wird nicht gecached. Das Resultat von USER wird gecached, sofern nicht innerhalb des php-Scriptes die Funktion $GLOBALS["TSFE"]->set_no_cache() aufgerufen wird.

USER / USER_INT benötigen auf höchster Objektebene eine Angabe der zu inkludierenden Datei (includeLibs).

Eigenschaft	Typ	Beschreibung
userFunc	Name der Funktion	userFunc gibt die gewünschte Funktion (z.B. in einer inkludierten Klasse) an. Beispiel: includeLibs.class1=fileadmin/class1.php.inc seite = PAGE seite.typeNum = 0 seite.10 = USER seite.10.userFunc = klassenname->main seite.10.meinObjekt = TEMPLATE
[String-array]		An die eigene php-Funktion werden die Eigenschaften unterhalb von „USER“ übermittelt. So kann z.B. aus der eigenen PHP-Funktionen der gesamte Abschnitt „meinObjekt“ gerendert und zurückgeliefert werden.

8.4.16 PHP_SCRIPT / PHP_SCRIPT_INT

Ähnlich wie USER / USER_INT.
PHP_SCRIPT / PHP_SCRIPT_INT kann keine Funktionen innerhalb der inkludierten Datei aus Typo3 heraus direkt ansprechen. Daher ist PHP_SCRIPT insbesondere für kleine Aufgaben geeignet.

Eigenschaft	Typ	Beschreibung
file	Resource / Datei	Datei, die inkludiert wird.

8.5 Frames

8.5.1 FRAME		
Eigenschaft	Typ	Beschreibung
obj	String	Hier wird angegeben, welche Seite in das jeweilige Frameset geladen werden soll (Name des PAGE-Objektes)
options	String	Hier können zusätzliche URL-Paramter angegeben werden.
params	String	Zusätzliche Frame-Parameter Beispiel: .params = scrolling="AUTO" noresize

8.5.2 FRAMESET		
Eigenschaft	Typ	Beschreibung
1,2,3,4,....	Fortlaufende Nummern	Hier werden die einzelnen Frameseiten definiert.
cols	String	Übliche HTML-Angabe der cols (siehe Beispiel unten)
rows	String	Übliche HTML-Angabe der rows (siehe Beispiel unten)
params	String	Die Parameter eines FrameSets. Beispiel: .params = border="0"

8.5.3 Beispiel-Framedefinition

```
unten = PAGE
unten.typeNum = 1

oben = PAGE
oben.typeNum = 3

frameset = PAGE
frameset.typeNum = 0
frameset.frameSet {
   rows = 150,*
   params = border="0" framespacing="0" frameborder="NO"

   1 = FRAME
   1.obj = oben
   1.params = scrolling="NO" marginwidth="0" marginheight="0"
   2 = FRAME
   2.obj = seite
   2.params = scrolling="Auto" noresize frameborder="NO"
}
```

8.6 OptionSplit

OptionSplit kann einer Eigenschaft mehrere Werte zuweisen, die sich z.B. abwechseln (alternieren) oder aber in Bereiche aufteilen (Anfang, Mitte, Ende).

|*|	Teilt einen Wert in die Bereiche Anfang, Mitte und Ende auf (Teilbereiche) Beispiel: seite.10.1 = GMENU seite.10.1.XY = 100,20 seite.10.1.backColor = blue |*| red |*| yellow Die Hintergrundfarbe ist für das erste Element blau, für das letzte Element gelb und alle Elemente in der Mitte rot.
||	Teilt einen Teilbereich (|*|) in erstes, zweites, drittes,... Element auf. Beispiel: seite.10.1 = GMENU seite.10.1.XY = 100,20 seite.10.1.backColor = blue || green |*| red |*| silver || yellow Hier wird erhält erste Element eine blaue, das zweite Element eine grüne Hintergrundfarbe. Alle Elemente in der Mitte erhalten eine rote Hintergrundfarbe. Das vorletzte Element erhält eine graue, das letzte Elemente eine gelbe Hintergrundfarbe.

8.7 Conditions

8.7.1 Browser	
[browser = …]	
Microsoft Internet Explorer	msie
Netscape Communicator	netscape
Lynx	lynx
Opera	opera
PHP fopen	php
AvantGo	avantgo
Adobe Acrobat WebCapture	acrobat
IBrowse (Amiga)	ibrowse
Teleport Pro	teleport
sonstige	unknown
Bei der Überprufung wird intern die Versionsnummer mit übergeben. Ist der Browser z.B. ein Netscape Communicator Version 4.72, lautet der zu überprüfende Browsername „netscape4.72“. Die Überprüfung auf einen Teilstring reicht hier aber aus. So würden z.B. „net“ oder „scape“ oder auch „netscape4“ hier anschlagen. Beispiel: [browser = netscape, opera]	

8.7.2 Browser-Version
[version = ...]
Beispiele: Nur für Versionen, die genau 4.03 sind: [version = 4.03] Nur für Versionen größer als 4 und ebenfalls für netscape-Browser Version 3 [version = > 4] [browser=netscape3]

8.7.3 Betriebssystem	
[system = ...]	
Windows 3.11	win311
Windows NT	winNT
Windows 95	win95
Windows 98	win98
Macintosh	mac
Linux	linux
SunOS	unix_sun
HP-UX	unix_hp
SGI / IRIX	unix_sgi
Amiga	amiga
Beispiel: [system = win, mac]	

8.7.4 Devices	
[device = ...]	
HandHeld / PDAs	pda
WAP-Handies	wap
Grabbers	grabber
Indexing robots	robot

8.7.5 Sprache
[language = ...]
Der Wert muß exakt mit dem Wert übereinstimmen, der in getenv("http_ACCEPT_LANGUAGE") in PHP gespeichert ist.

8.7.6 IP-Adressen
[IP =...]
Beispiele: [IP = 123.*.*.*] [IP = [192.168.*.*] [62.153.151.126]

8.7.7 Stunde
[hour = ...]
Beispiele: [hour = > 17]

8.7.8 Minute
[minute = …]
Mögliche Werte 1..59 wie “Stunde”.

8.7.9 Wochentag
[dayofweek = …]
Mögliche Werte 0..6 Sonntag = 0 bis Samstag = 6 Beispiele: [dayofweek =0]

8.7.10 Tag des Monats
[dayofmonth = …]
Mögliche Werte 1..31 Beispiel: [dayofmonth = > 25]

8.7.11 Monat
[month = …]
Mögliche Werte 1..12 Beispiel: [month = 12]

8.7.12 Benutzergruppe (FE)
[usergroup = …]
Bei usergroup wird die uid der gewünschten FE-Benutzergruppe angegeben Beispiel: [usergroup = 16]

8.7.13 Eingeloggter Benutzer (FE)
[loginUser = …]
Bei loginUser wird die uid des gewünschten FE-Benutzers angegeben Beispiele: Findet genau einen FE-User: [loginUser = 25] Findet alle eingeloggten FE-Benutzer: [loginUser = *]

8.7.14 treeLevel
[treeLevel = …]
Prüft auf die aktuelle Ebene innerhalb des Baumes. Insb. für Manipulationen bei der Template-Vererbung sinnvoll einsetzbar. Beispiel: [treeLevel = 3]

8.7.15 PIDInRootline
[PIDInRootline = …]
Prüft, ob sich eine bestimmte Seite unterhalb einer bestimmten Seite befindet. Insb. für Manipulationen bei der Template-Vererbung sinnvoll einsetzbar. Beispiel: [PIDInRootline = 12, 216]

8.7.16 PIDupinRootline
[PIDupinRootline = …]
Wie „PIDInRootline“. Die aktuelle Seite wird hier jedoch ausgeschlossen.

8.7.17 GlobalVar / GlobalString
[globalVar = …] **[globalString = …]**
Proft auf globale Variablen (GET, POST, Umgebungs-Variablen) ENV, GP, TSFE, LIT Beispiele: [globalString = HTTP_HOST=www.typo3server.com] [globalString = ENV:REMOTE_ADDR=192.168.*.*] [globalVar = TSFE:id > 10]

8.7.18 userFunc
[userFunc = <Funktions-Name>]
Hier kann eine eigene PHP-Funktion angegeben werden, die true bzw. false zurückliefert. Die PHP-Funktion selbst wird in der Datei localconf.php abgelegt. Beispiel: [userFunc = meineFunktion()]

8.8 Primäre Objekte

8.8.1 PAGE		
Eigenschaft	Typ	Beschreibung
Nähere Informationen unter „Objekt-Referenz -> PAGE“		

8.8.2 CONFIG		
Eigenschaft	Typ	Beschreibung
linkVars	stringliste	Variablen aus HTTP_GET_VARS, die immer an eine erzeugte URL angefügt werden soll. Beispiel: config.linkVars = session Dies fügt an einen Link den Parameter „session“ mit einem übertragenem Wert. Hinweis: Einige Module / Erweiterungen erzeugen direkte Links, hierdurch kann diese Eigenschaft ggf. unbrauchbar werden.
message_page_is_being_generated	string	Alternativer Nachricht (HTML) zu „Page is being generated“.
message_preview	string	Alternativer Text (HTML) die erscheinen soll, wenn die Preview-Funktion aktiviert ist. Default: PREVIEW
locale_all	string	PHP-setlocal-Funktion Nähere Informationen unter php.net Beispiel: config.local_all = de_DE
spamProtectEmailAddresses	boolean	Wenn diese Eigenschaft gesetzt wird, werden alle eMail-Adressen verschlüsselt, so dass eMail-Robots diese eMail-Adresse nicht mehr entschlüsseln können.
spamProtectEmailAddresses_atSubst	string	Alternative Angabe des @-Zeichens (Default ist (at), sofern spamProtectEmailAddresses aktiviert ist). Default: (at)
frameReloadIfNotInFrameset	boolean	Wenn diese Eigenschaft aktiviert ist, muss immer das gesamte Frameset geladen sein – einzelne Frameseiten lassen sich nicht separat im Browser öffnen. Hinweis: Die typeNum muss ungleich 0 sein!
includeLibrary	resource	Hier kann eine PHP-Datei inkludiert werden.

cache_periode	int (sec)	Hier kann in Sekunden angegeben werden, wie lange eine Seite im Cache verweilen soll. Hinweis: Der Wert dieser Eigenschaft kann durch jede angelegte Seite überschrieben werden. Default: 86400 (=24 Std.)
cache_clearAtMidnight	boolean	Wird diese Eigenschaft aktviert, verfällt der Cache einer Seite Nachts um 0 Uhr. Default: false
no_cache	boolean	Wenn diese Eigenschaft aktiviert ist, wird kein Cache verwendet. Hinweis: Diese Eigenschaft sollte aus Performance-Gründen nicht aktiviert werden! Default: false
stat	boolean	Typo3-Interne Statistiken werden geführt. Default: true
stat_excludeBEuserHits	boolean	Wenn diese Eigenschaft aktiviert wird, werden Seitenzugriffe von eingeloggten Backend-Benutzer nicht mit protokolliert.
stat_excludeIPList	string	Wenn die IP-Adresse (REMOTE_ADDR) in diesem String enthalten ist, werden die Seitenzugriffe ebenfalls nicht mit protokolliert.
stat_mysql	boolean	Aktiviert das Schreiben von Logs in die DB-Tabelle sys_stat. Default: false
stat_apache	boolean	Aktiviert das Schreiben von Logs in eine unter „stat_apache_logfile“-angegebenen Datei.
stat_apache_logfile	filename	Hier wird angegeben, wie die Datei heißt, in die die Logs geschrieben werden. Hinweis: Diese Datei muss vorhanden und schreibbar sein. Hinweis: Keine Pfadangabe! In der Datei localconf.php wird unter $TYPO3_CONF_VARS[“TSFE“][“logfile_dir“] der Pfad zur Logdatei angegeben.
stat_apache_pagenames	string	Hier kann die Simulierung des Dateinames definiert werden. Default ist [path][title]—[uid].html Mögliche Werte sind:

		[title], [uid], [alias], [type], [path]
simulateStaticDocuments	boolean / PATH_INFO	Mit dieser Option können statische Seiten simuliert werden. Von Typo3 aus erzeugt Links verweisen nicht mehr auf index.php?id=123, sondern z.B. auf 123.0.html [uid].[type].html bzw. [alias].[type].html Hierzu muss eine .htaccess-Datei existieren, die z.B. folgenden Inhalt hat: RewriteEngine On RewriteRule ^[^/]*\.html$index.php Mit PATH_INFO können Windows-Benutzer die PHP-Funktionen nutzen.
simulateStaticDocuments_addTitle	boolean	Wenn diese Eigenschaft aktiviert wird, wird zusätzlich der Title der Seite vorangestellt. [title].[uid].[type].html
simulateStaticDocuments_noTypeIfNoTitle	boolean	Sofern der Type 0 ist, wird dieser nicht mit in den Link aufgenommen. Aus [uid].[type].html bzw. [alias].[type].html wird somit [uid].html bzw.[alias].html
simulateStaticDocuments_pENC	string	Kodiert zusätzliche Parameter (z.B. index.php?id=123&tt_news=1234 Mögliche Werte sind: base64, md5 Vor- und Nachteile von base64: Keine Parameter in der URL Die URL wird sehr lang Probleme mit Suchmaschinen Vor- und Nachteile von md5: Die URL wird kürzer als bei base64 Der Dateiname wird in einer Caching-Tabelle gehalten. Hierdurch können Caching-Probleme auftreten!
titleTagFunction	Funktionsname	Um den Titel der Seite manuell zu setzen, kann eine Funktion hierzu aufgerufen werden (anzulegen in der localconf.php)
headerComment	string	Hier kann ein beliebiger Kommentar angegeben werden, der vor dem Typo3--Hinweis im Quelltext erscheint.
notification_email_encoding	string	Hier kann für versendete eMails (plaintext) der jeweilige encode-Modus angegeben werden. Mögliche Werte sind: base64, quoted-printable, 8bit
notification_email_urlmode	string	Bei versendeten eMails können Links lang

		werden. Um einen Zeilenumbruch zu verhinden, werden „Shortcuts“ innerhalb einer Tabelle gesetzt, die auf den Originallink zeigen. Mögliche Werte sind: [empty], 76, all
admPanel	boolean / ADM-Panel	Wenn admPanel aktiviert ist, wird (bei eingeloggtem Backend-User) ein Admin-Panel angezeigt. Hinweis: Der Admin-Panel muss für jeden Benutzer separat im Feld TSConfig aktiviert werden!
index_enable	boolean	Gecachete Seiten dürfen indiziert werden.
index_externals	boolean	Wenn diese Eigenschaft aktiviert ist, werden auch „externe“ Media-Links (z.B. PDF, DOC) mit indiziert.
sys_language_uid	String	Setzt die jeweilige Sprach-ID (php), wie z.B. "de".

8.8.3 FE_DATA / FE_TABLE		
Eigenschaft	Typ	Beschreibung

[table].default.[field]	string	Hier wird angegeben, welche Werte für neue Datensätze „Default“ sind.
		Beispiel:
		.tt_content.default {
		hidden = 1
		CType = 0
		header = Dies ist die Überschrift
		}

[table].allowNew.[field]	string	Hier wird angegeben, in welche Felder bei der Neuanlage eines Datensatzes aus dem Frontend heraus geschrieben werden darf. Beispiel: .tt_content.allowNew.hidden = 0 .tt_content.allowNew.CType = 0 .tt_content.allowNew.title = 1
[table].allowEdit.[field]	string	s.o.
[table].autoInsertPID	boolean	Bei aktivierter Eigenschaft wird die pid bei neuen Datensätzen automatisch hinzugefügt.
[table].processScript	resource	Include-Script, dass die Daten in die Datenbank schreibt. Als Beispiel ist das Script des Gästebuch-Moduls empfehlenswert, zu finden unter typo3/ext/tt_guest/pi/guest_submit.inc
[table].doublePostCheck	string (Feldname)	Gibt einen Feldnamen (Integer) an, in dem ein Integer-Hash-Wert gespeichert wird. Hierdurch können doppelte Einträge vermieden werden.

8.9 TSConfig : Benutzer

8.9.1 admPanel		
Hinweis: Damit das Adminpanel angezeigt und aktiviert wird, muss ebenfalls in TypoScript config.admPanel aktiviert werden.		
Eigenschaft	Typ	Beschreibung
enable.[bereich]	boolean	Hier können verschiedene Bereiche des Adminpanels aktiviert werde. Folgende Bereiche stehen zur Verfügung: all (aktiviert alle folgenden Bereiche) preview, cache, publish, edit, tsdebug, info Beispiel: admPanel.enable.edit = 1 Hinweis: Für Admin-User muss der admPanel nicht explizit gesetzt werden. Hier ist admPanel.enable.all = 1 immer gesetzt.
hide	boolean	Wenn hide aktiviert ist, wird das Adminpanel nicht angezeigt.
[module].edit	boolean	Folgende Edit-Bereiche können hier explizit gesetzt werden: forceDisplayIcons forceDisplayFieldIcons forceNoPopup Beispiel: admPanel.forceNoPopup.edit = 1

8.9.2 options		
Eigenschaft	Typ	Beschreibung
RTEkeyList	String (komma-separiert)	Hier kann der Richtext-Editor für den Benutzer konfiguriert werden. Folgende Elemente können mit in die kommaseparierte Liste aufgenommen werden: cut, copy, paste, formatblock, fontstyle, fontsize, bold, italic, underline, left, center, right, orderedlist, unorderedlist, outdent, indent, line, link, table, image, textcolor Hinweis: Erhält options.RTEkeyList keinen Wert zugewiesen, werden alle Elemente angezeigt.
clearCache.pages	boolean	Der Benutzer kann den gesamten Cache einer

		Seite löschen
clearCache.all	boolean	Der Benutzer kann alle Caches löschen
lockToIP	String (komma-separiert)	Stringliste mit möglichen IP-Adressen, von denen sich der Benutzer einloggen kann. Wildcards (*) sind erlaubt. Beispiel: options.lockToIP = 192.*.*.*, 62.4.*.*
saveDocNew saveDocNew.[table]	boolean	Mit aktiviertem saveDocNew wird für den Benutzer ein zusätzlicher Button „Speichern und neuen Datensatz anlegen“-Button angezeigt. Mit [table] kann diese Option für ganz bestimmte Module aktiviert werden. Beispiel: saveDocNew = 0 saveDocNew.tt_products = 1
disableDelete disableDelete.[table]	boolean	Deaktiviert den “Löschen”-Button. Mit [tables] kann explizit der Button für eine bestimmtes Modul gesteuert werden.
contextMenu.[key].disableItems	String (komma-separiert)	Hier können die einzelnen Elemente aus den Klickmenüs (Element-Browser) deaktiviert werden. key kann folgende Wert annehmen; pageTree, pageList, folderTree, folderList Mögliche Objekte für „page“-Werte: view, edit, hide, new, info, copy, cut, paste, delete, move_wizard, history, perms, new_wizard, hide, edit_access, edit_pageheader, db_list Mögliche Objekte für “folder”-Werte: edit, upload, rename, new, info, copy, cut, paste, delete Beispiel: contextMenu.pageTree.disableItems = delete

8.9.3 setup		
Setup hat zwei Unterwerte: default und override. Sämtliche hier aufgeführten Eigenschaften können somit setup.default... oder setup.override... sein. Default gibt an, welche Werte beim ersten Login des Benutzers gelten sollen. Override gibt an, dass die zum Benutzer gespeicherten Werte überschrieben werden sollen.		
Eigenschaft	Typ	Beschreibung
thumbnailsByDefault	boolean	Thumbnails werden angezeigt.
startInTaskCenter	boolean	Wenn diese Eigenschaft aktiviert wird, öffnet sich das Backend im „TaskCenter“.
saveTreePositions	boolean	Wenn diese Eigenschaft aktiviert wird, merkt sich Typo3 beim Logout die zuletzt bearbeitet Seite und öffnet diese beim nächsten Login.
edit_RTE	boolean	Aktiviert bzw. deaktiviert den RichText-Editor.
copyLevels	int (positiv)	Anzahl der Ebenen, die ein Benutzer rekursiv kopieren darf.
recursiveDelete	boolean	Aktiviert das rekursive Löschen. Alle Unterseiten und deren Inhalte werden mitgelöscht.

8.10 TSConfig : Page

8.10.1 mod		
Eigenschaft	Typ	Beschreibung
web_layout.menu.function		Modul: Web > Page 0 => QuickEdit 1 => Columns 2 => Languages
web_info.menu.function		Modul : Web > Info page => Pagetree overview log => Admin Changelog tsconf => Page TSconfig
web_func.menu.function		Modul: Web>Functions 1 => Import 2 => Export 3 => Wizards
web_ts.menu.function		Modul: Web>Template 0 => Info/Modify 1 => Constant editor 2 => TypoScript Object browser 3 => Template analyzer
user_task.menu.function		Modul: User>Task center "tasks" => Tasks "messages" => Messages "note" => Quick Note "recent" => Recent Pages "mod" => Modules "actions" => Actions
SHARED.colPos_list		Spezielle Eigenschaft, mit der sich die Content-Spalten „Links, Normal, Rechts, Rand“ aktivieren bzw. deaktivieren lassen. “0” => Normal “1“ => Left “2“ => Right “3“ => Border Beispiel: mod.SHARED.colPos_list = 0,2

[mod].tt_content.colPos_list		Eigenschaft, mit der sich die Content-Spalten „Links, Normal, Rechts, Rand“ aktivieren bzw. deaktivieren lassen. “0” => Normal “1“ => Left “2“ => Right “3“ => Border Beispiel: mod.web_layout.tt_content.colPos_list = 0,2
[mod].tt_content.fieldOrder	string (komma-separiert)	Sortierreihenfolge im Backend für tt_content-Elemente.
[mod].noCreateRecordsLink	boolean	Der Link „Neuen Datensatz anlegen“ wird nicht angezeigt.
[mod].alternateBgColors	boolean	Mit aktivierter Eigenschaft werden Hintergründe bei Elementen alternierend dargestellt.
newWizards	boolean	Aktiviert bzw. deaktiviert den „Neuen-Datensatz-Anlegen“-Wizard.

Anhang

Rückmeldungen

Auf Anregungen, Kritik und Lob zur Deutschen Typo3-Dokumentation freue ich mich unter der eMail-Adresse dokumentationen@typo3server.com.

Interessante Links

- Diese Dokumentation befinden sich ständig in der Weiterentwicklung. Den aktuellen Stand können Sie sich unter http://www.typo3server.com/dokumentationen.html herunterladen.

- Bei Fragen zu Typo3 (Probleme mit der Entwicklung):
  Das Deutsche Typo3-Forum:
  http://www.typo3.net

- Informationen zu Typo3-Schulungen:
  http://www.typo3server.com/workshop.html

Typo3 kostenlos testen

Wir empfehlen zum Testen und Entwickeln das Typo3-Testpaket, das für einen Monat kostenlos ohne weitere Verpflichtungen zur Verfügung gestellt wird. Informationen erhalten Sie unter: http://www.typo3server.com/typo3_testen.html

Realisierung

Bei Fragen zu Realisierbarkeiten etc. stehen wir Ihnen Montags bis Freitags von 9:00 bis 16:00 Uhr zur Verfügung: Tel.: 05772 / 9116-70.