MITTWALD
WEBHOSTING · SCHULUNGEN · SUPPORT

Grundlagenwissen TYPO3 Version 4.2.x

Deutsche TYPO3-Dokumentation

Mittwald CM Service GmbH und Co. KG
Königsberger Straße 6
32339 Espelkamp

URL: http://www.mittwald.de
E-Mail: t3doku@mittwald.de

Inhaltsverzeichnis

1. Vorwort

Diese Dokumentation ist eine Neuauflage der deutschen TYPO3-Dokumentation (TYPO3 – die deutsche Dokumentation mit Referenz). Die durchweg positive Resonanz auf die Vorgängerversionen hat mich darin bestärkt, das Konzept beizubehalten. Eine inhaltliche Aktualisierung ist notwendig, da TYPO3 in der Version 4.x wesentliche Änderungen beinhaltet.

Die deutsche TYPO3-Dokumentation soll in erster Linie Grundlagenwissen vermitteln, um den Einstig in die komplexe Software zu ermöglichen.

Auf Modulkonfigurationen wird beispielsweise nur bedingt eingegangen, jedoch 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 E-Mail-Adresse **t3doku@mittwald.de**.

Für die aktive Unterstützung bei der Überarbeitung der vorliegenden Dokumentation möchte ich an dieser Stelle meinem Kollegen Olaf Clemens herzlich danken.

Ich wünsche Ihnen viel Spaß beim Erlernen von TYPO3.

Ihr Robert Meyer

2. Schritte zur ersten Seite

2.1 Voraussetzungen

Für die Arbeit mit der Dokumentation wird ein installiertes TYPO3 vorausgesetzt. Wie TYPO3 installiert wird und welche Vorrausetzungen benötigt werden, ist nicht Bestandteil dieser Dokumentation. Informationen zur Installation und den Systemvoraussetzungen finden Sie unter:

http://www.typo3.org/documentation/document-library/

Ich empfehle Ihnen, eine Umgebung bei einem auf TYPO3 spezialisierten Hostinganbieter zu nutzen. In der Regel können Sie dort ein kostenloses, jedoch zeitlich begrenztes Testpaket buchen.

Alternativ gibt es für den lokalen nicht produktiven Einsatz ein vorkonfiguriertes Installations-Paket, welches einen Webserver, ein Datenbanksystem und benötigte Komponenten für Windows, Linux und Macsysteme enthält. Diese sogenannten „Installer“-Pakete stehen unter dem folgenden Link zur Verfügung:

http://www.typo3.org/download/installers/

2.2 Begrifflichkeiten

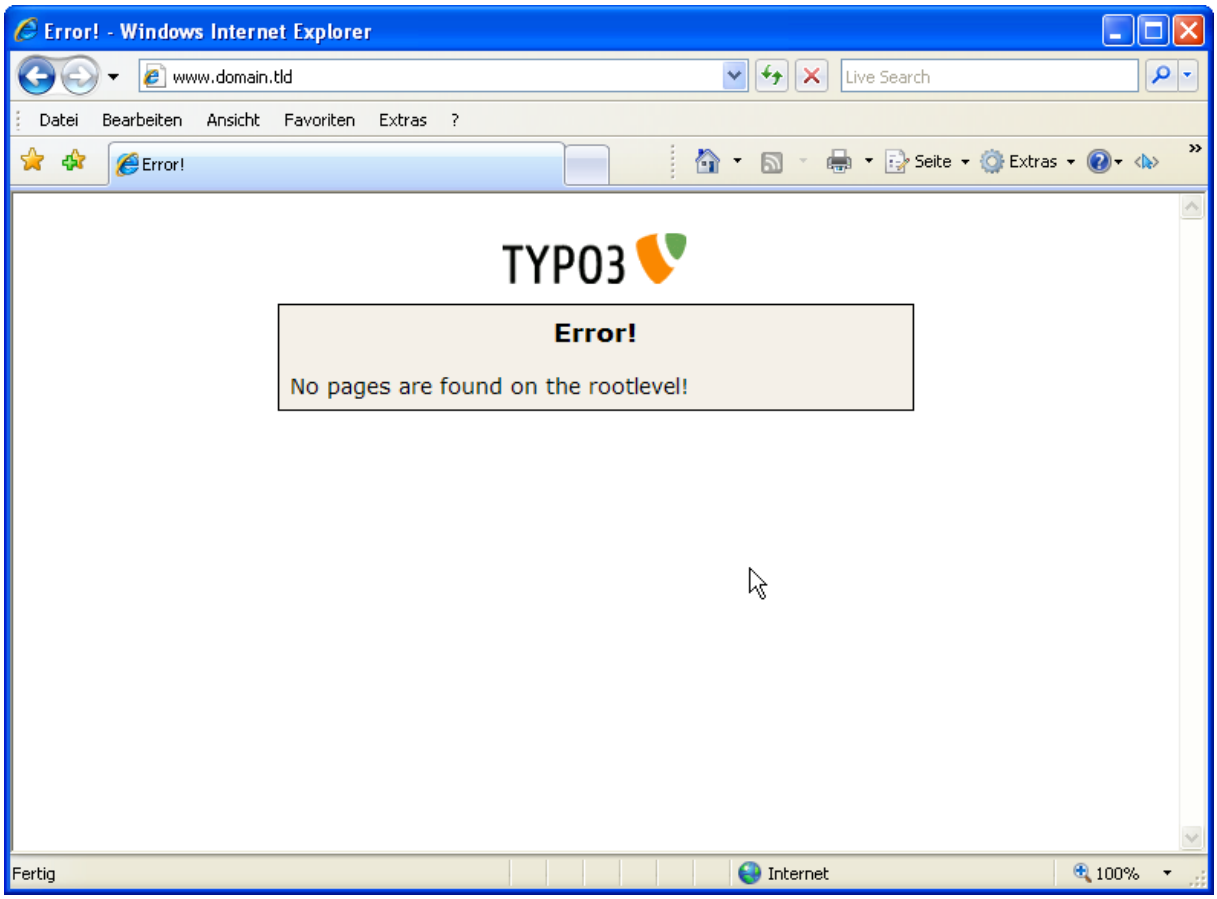
Da TYPO3 überwiegend in Dänemark entwickelt wird und die gesamte Dokumentation überwiegend in englischer Sprache verfügbar ist, versuchen wir hier, einige deutsche Begriffe einzuführen.

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 ihren 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, dass 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 bzw. Templates für TypoScript-Templates verwendet.

2.3 Erster Frontend-Aufruf

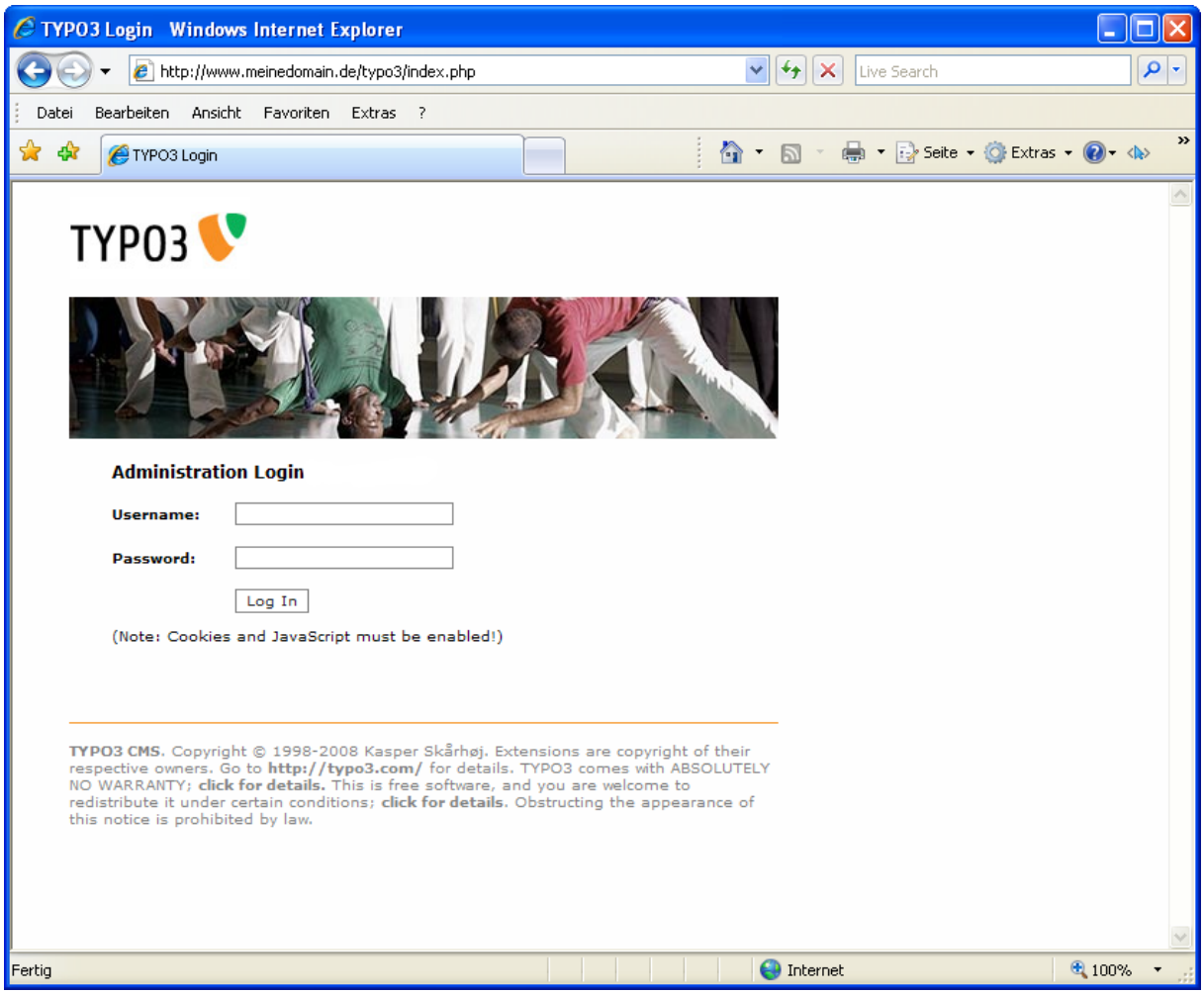
Vorausgesetzt TYPO3 ist korrekt und einwandfrei installiert, erhalten wir beim einfachen Aufruf der Internetpräsentation bei einem „leeren und sauberen“ TYPO3 folgende Fehlermeldung:



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

2.4 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:

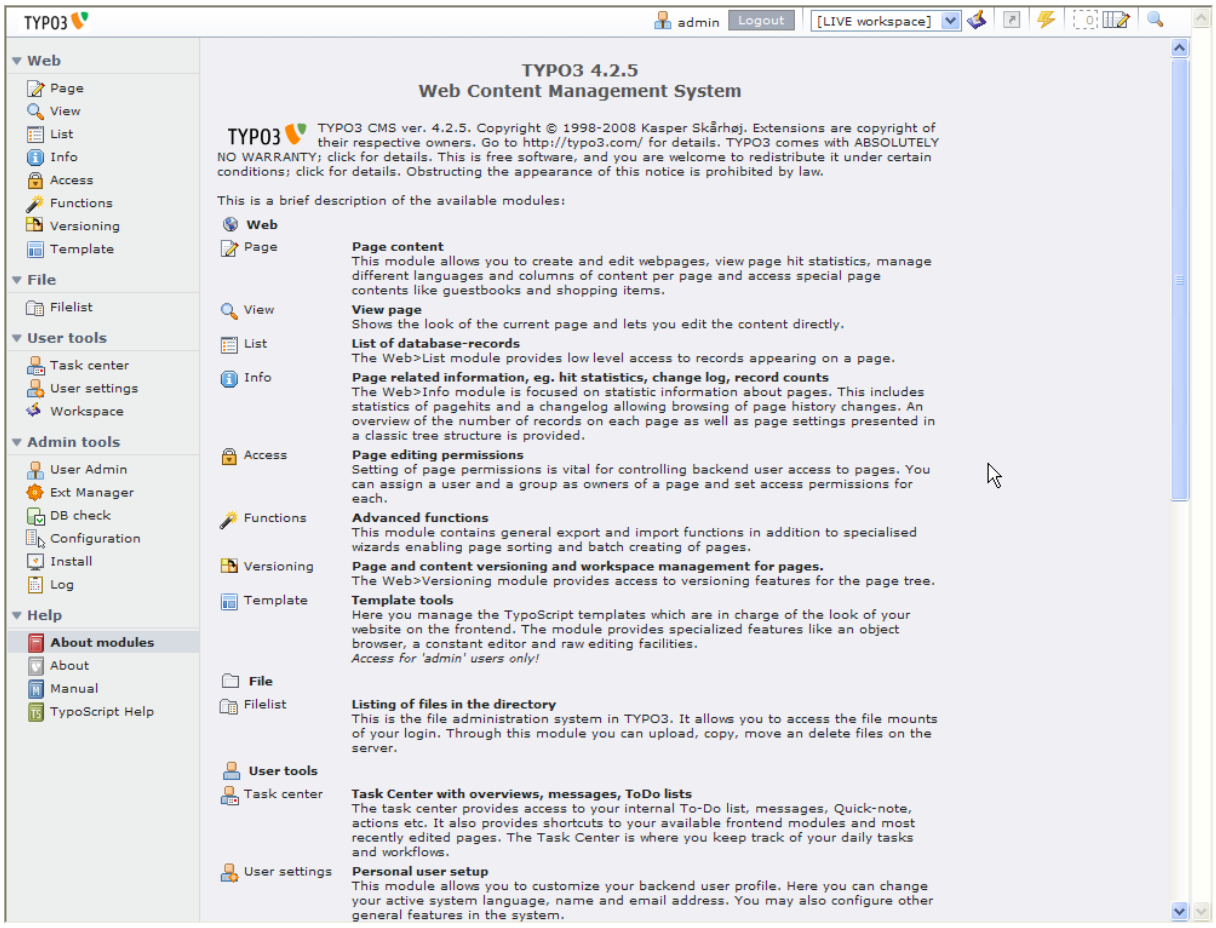


Hinweis:
Zum erfolgreichen Login in das Backend ist es erforderlich, dass Cookies auf dem jeweiligen Client aktiviert sind. Ebenfalls kann lokal auf Ihrem PC installierte AntiViren- und Firewallsoftware den Login-Vorgang verhindern, da diese, je nach Konfiguration, den „Referer“ nicht mit übergeben. Sollten Sie eine Nachricht der Art erhalten, dass Sie im Installtool einen „*doNoCheckReferer*“-Flag setzen sollen, loggen Sie sich in das Install-Tool unter **www.meinedomain.de/typo3/install** ein. Das Passwort wurde Ihnen von Ihrem Anbieter mitgeteilt oder es ist das Standard-Passwort **joh316** gesetzt. Im **Menüpunkt 5: "All Configuration"** können Sie das Flag setzen.

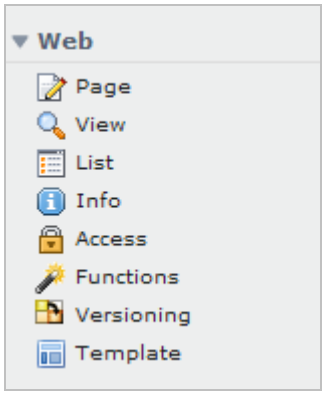
Es kann durchaus der Fall sein, dass es noch eine dritte Zeile gibt, in der wir über eine Auswahlbox das Interface auswählen können. Mögliche Oberflächen können sein „Frontend“ oder „Backend“. Wählen Sie bei der Anmeldung „Backend“ aus.

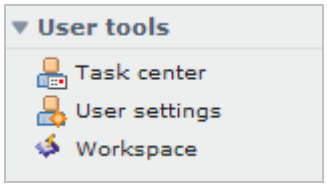
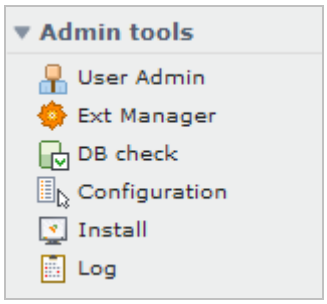
2.5 Kurzer Überblick über die Backend-Module

Nach erfolgtem Login als Administrator gelangen Sie in das TYPO3 Backend.



Im Folgenden werden die einzelnen Backend-Modulbereiche vorgestellt:

Modul	Englisch	Deutsch	Beschreibung
Web			Unter dem Web-Modul wird redaktionell gearbeitet. Ein Seitenbaum wird dargestellt.
	Page	Seite	Hier finden der strukturelle Aufbau sowie die Pflege der Internetseiten statt. Administratoren legen die Struktur fest, Redakteure können Inhalte bearbeiten.
	View	Ansicht	Zeigt die Präsentation an. Auf Grund der ungünstigen Darstellung im TYPO3 Backend zu vernachlässigen.
	List	Liste	Hier werden sämtliche Datensätze installierter Module angezeigt, die sich auf einer bestimmten Seite befinden. Ähnlich wie „Seite“, jedoch im Listen Format.
	Info	Info	Zu vernachlässigen. Häufig wird das Modul von TYPO3 Erweiterungen genutzt und bietet dann zusätzliche Informationen an.
	Access	Zugriff	Einzelne Seiten können mit Zugriffsrechten (Besitzer, Gruppe,

			Alle: Lesen, Schreiben, Löschen) versehen werden. Wichtiges Modul, wenn mit Redakteuren und Benutzergruppen gearbeitet werden soll.
	Functions	Funktionen	Nützliche Tools wie z.B. das Anlegen von bis zu 10 Unterseiten auf einmal.
	Versioning	Versionen	Über das Modul ist eine Versionierung von Seiten und Seiteninhalten möglich.
	Template	Template	Der wohl wichtigste Bereich: Hier wird mit TypoScript gearbeitet. Zusätzlich stehen Tools wie z.B. der Objekt-Browser zur Verfügung.
File			
	Filelist	Dateiliste	Ein kleines online-„FTP-Programm“ mit beschränkten Möglichkeiten. Für einfache Uploads bzw. Änderungen jedoch gut geeignet. Bitte beachten Sie den Hinweis „Dateiliste“.
User Tools			
	Task center	Aufgaben	Unter „Aufgaben“ kann ein Benutzer sich selbst oder anderen Benutzern Aufgaben zuordnen (benötigt die TYPO3 System Erweiterung „*sys_action*“). Oftmals vernachlässigbar, wird in der Regel durch das Workspace Konzept ersetzt.
	User Settings	Benutzer Einstellungen	Hier kann ein Benutzer seine Backend-Sprache einstellen ein eigenes Passwort vergeben und Informationen zur Person hinterlegen. Weitere Einstellungen gelten insbesondere der Darstellung des Backends.
	Workspace	Arbeits-umgebung	Ermöglicht das Arbeiten in verschiedenen Arbeitsbereichen, z.B. eine Vorschau und Live-Bereich. Über das Modul können Inhalte gewechselt, Live oder als Preview geschaltet werden.
Admin tools			
	User Admin	Benutzer-verwaltung	Bei der intensiven Arbeit mit unterschiedlichen Benutzern (Redakteuren) und unterschiedlichen Rechten ist das Modul „Benutzer Administrator“ eine gute Möglichkeit, Übersicht zu behalten. Weiter können Sie prüfen, welche Benutzer derzeit im System angemeldet sind.
	Ext Manager	Erweiterungs-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	Im Modul werden Werkzeuge zur Überprüfung der Datenbank zur Verfügung gestellt.
	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	Protokoll	Sämtliche Logins und Datensatz-Änderungen werden hier festgehalten. Bei Änderungen können diese oftmals rückgängig gemacht werden.

2.5.1 Hinweis zum Modul Filelist/Dateiliste

Mit diesem kleinen "FTP-Programm" können Dateien hochgeladen, modifiziert und gelöscht werden. Jedoch kann es hierbei zu Problemen mit Dateirechten kommen, da ebenfalls 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 Mittwald CM Service treten diese Probleme nicht auf. Hier werden die Dateien, die per FTP oder das TYPO3 Backend übertragen werden, mit den gleichen Benutzerrechten angelegt.

2.5.2 Sprache ändern

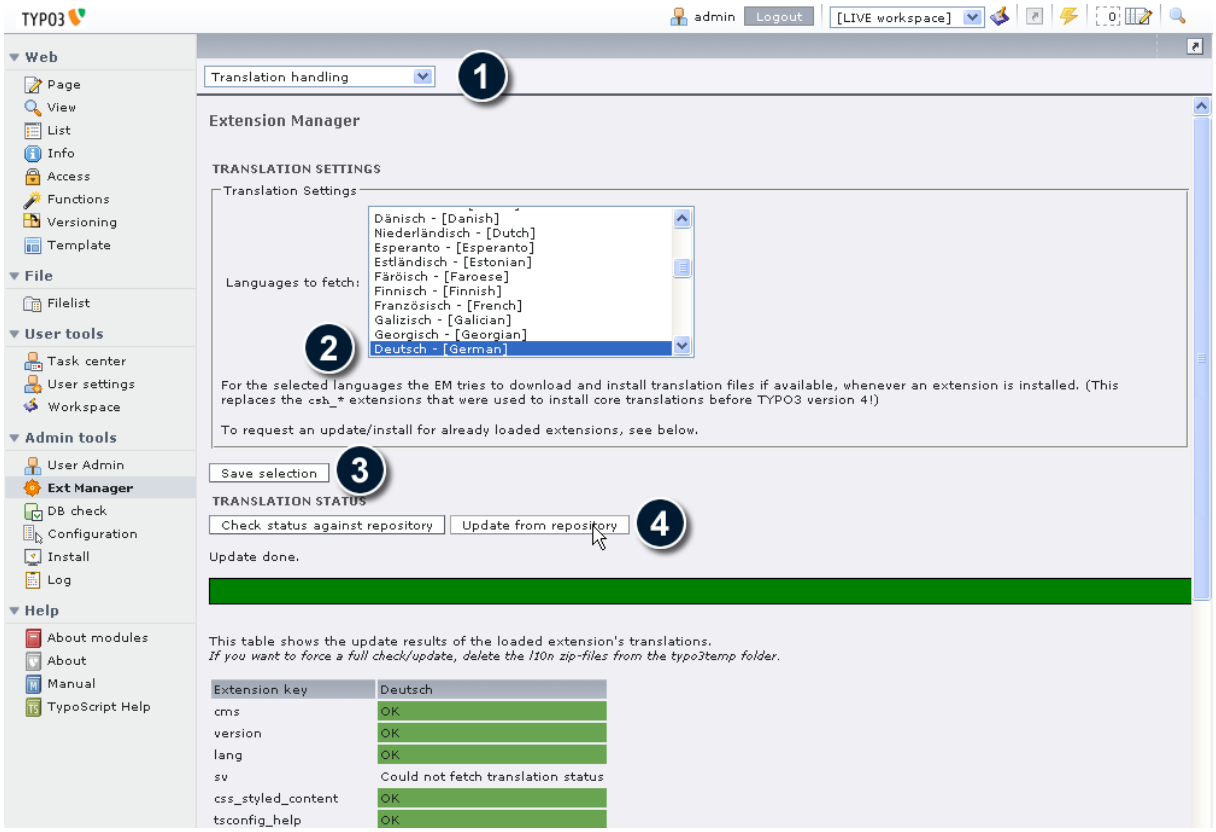
Die Standardsprache in TYPO3 ist Englisch. Die erste Aktion, die in der Regel bei einem englischsprachigen Backend ausgeführt wird, ist die Änderung der Sprache auf „Deutsch“.

Zur Umstellung der Sprache werden je nach Sprache unterschiedliche Sprachdateien benötigt, welche seit der TYPO3 Version 4.0 über den Extension-Manager installiert werden.

Nach dem erfolgreichem Backend Login gehen wir im Bereich „***Admin tools***“ auf das Modul ***„Ext Manager“***. Auf der rechten Seite wird im oberen Bereich eine Auswahlbox angezeigt, in der wir den Punkt „***Translation Handling***“ auswählen.

In der angezeigten Liste der zur Verfügung stehenden Sprach-Pakete wählen wir den Eintrag „***German – [German]***“ aus und bestätigen die Auswahl über die Schaltfläche „***Save selection***“.

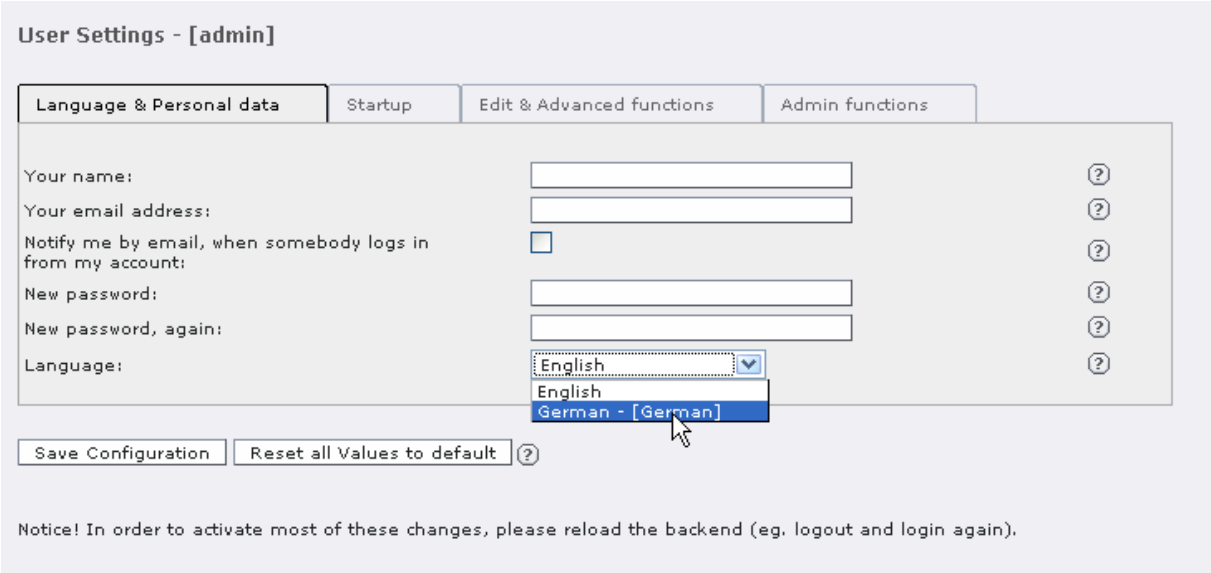
Der Download der Sprachdateien wird über die Schaltfläche „***Update from repository***“ gestartet.



Hinweis:
Sollte der Download fehlschlagen, prüfen Sie, ob TYPO3 Schreibrechte auf das Verzeichnis typo3*temp/* hat. In diesem werden die Sprachen als *l10n zip* Dateien abgelegt (z.B. beuser-l10n-de.zip). Weiter muss die PHP Einstellungen *allow_url_fopen* aktiviert sein und ausreichend *memory_limit* zur Verfügung stehen.

Nach dem Installieren der Dateien müssen wir die Sprache für unseren Benutzer aktivieren. Hierzu wählen wir auf der linken Seite aus dem Bereich „***User tools***“ den Menüpunkt „***User settings***“ aus.

Auf der rechten Seite erhalten Sie eine Ansicht verschiedener Registrierkarten. Auf dem Reiter „***Language & Personal data***“ befindet sich neben dem Feld „***Language***“ eine Auswahlbox für die Sprache.



Wir ändern hier die Sprache von „English“ auf „***German***“ ab und speichern die Einstellungen über die Schaltfläche „***Save Configuration***“.

Das Fenster wird automatisch neu geladen und ab sofort in deutscher Sprache angezeigt.

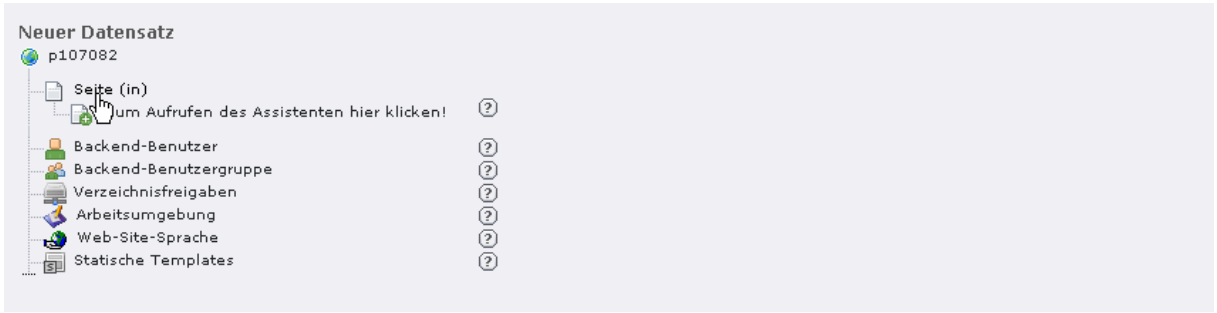
2.6 Eine erste Seite anlegen

Um eine erste Seite mit TYPO3 anzulegen, gehen Sie im Bereich „***Web***“ auf den Menüpunkt „***Seite***“.

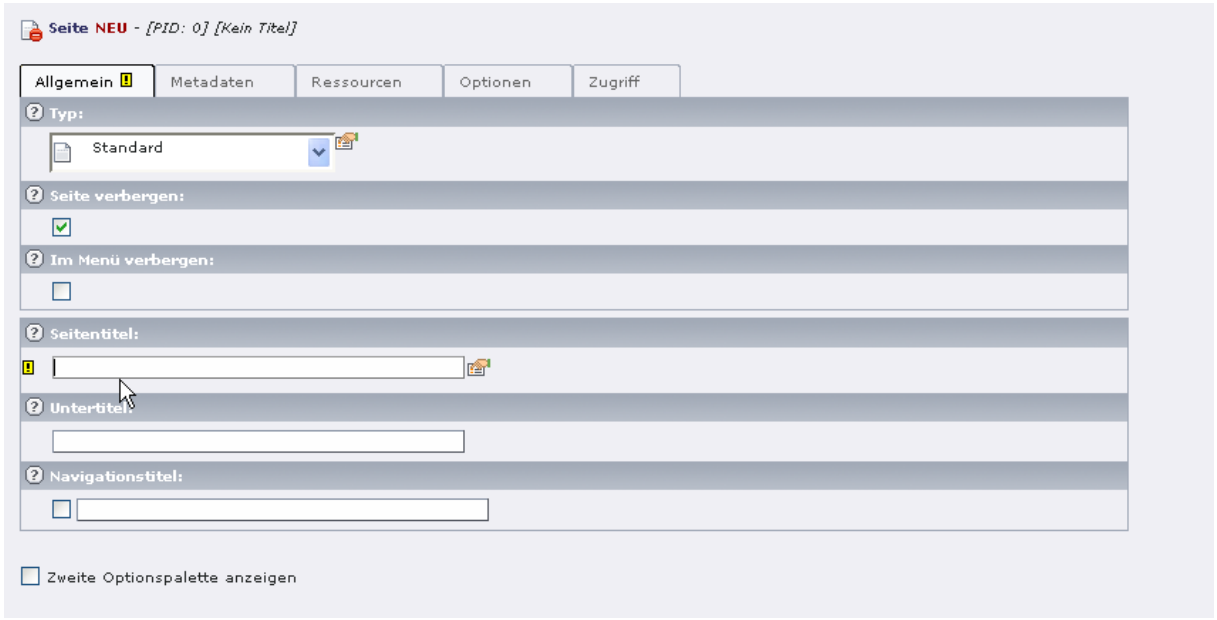
Es öffnet sich in der Mitte der Seitenbaum, der derzeit nur aus dem Rootlevel (Icon Weltkugel) besteht. Auf der rechten Seite öffnet sich nur ein kleiner Hinweis.

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 (in)***“ (oberster Eintrag).



Es öffnet sich rechts eine Maske, in der diverse Felder ausgefüllt werden können. Pflichtfelder werden mit einem gelben Ausrufungszeichen vor 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.

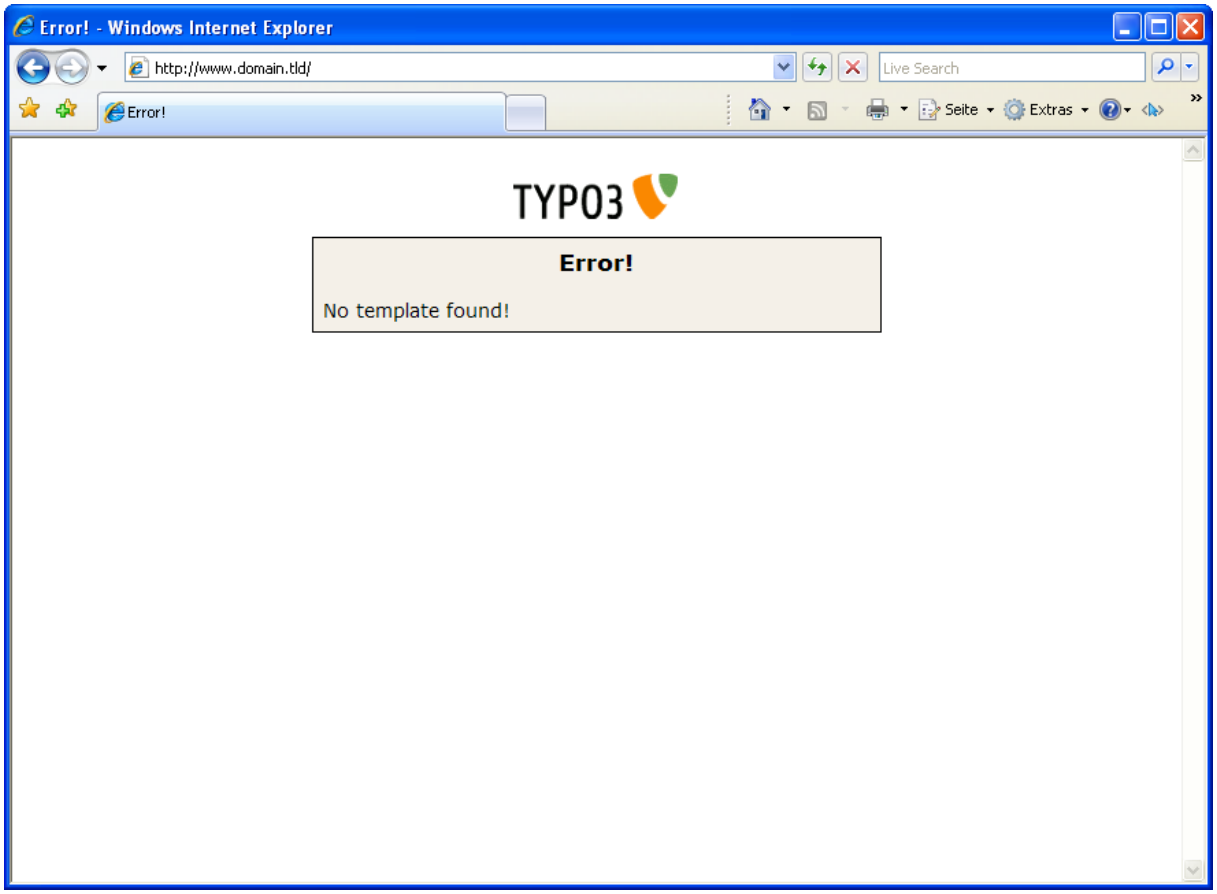


Hinweis:
Beim Anlegen einer Seite ist immer die Option „Seite verbergen“ aktiviert. Wenn Sie später Seiten sichtbar schalten möchten, müssen Sie das Häkchen entfernen.

3. Templates

Templates teilen TYPO3 mit, wie TYPO3 was zu machen hat. Ohne ein Template auf einer Seite oder auf einer übergeordneten Seite „weiß“ TYPO3 nicht, dass es überhaupt etwas im Frontend darstellen soll.

Ist keine Seite in TYPO3 angelegt, erhalten wir im Frontend die Meldung „Error! No pages are found on the rootlevel!“. Im Kapitel 2 haben wir bereits eine Seite „test“ angelegt, wodurch sich die Fehlermeldung änderte nach „Error! No template found!“.



TYPO3 kann die angelegte Seite „test“ nicht anzeigen, da keine Informationen in Form eines Templates hinterlegt worden sind. Somit fehlen die benötigten Informationen, wie eine Seite angezeigt werden soll. Es wird also ein Template benötigt.

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

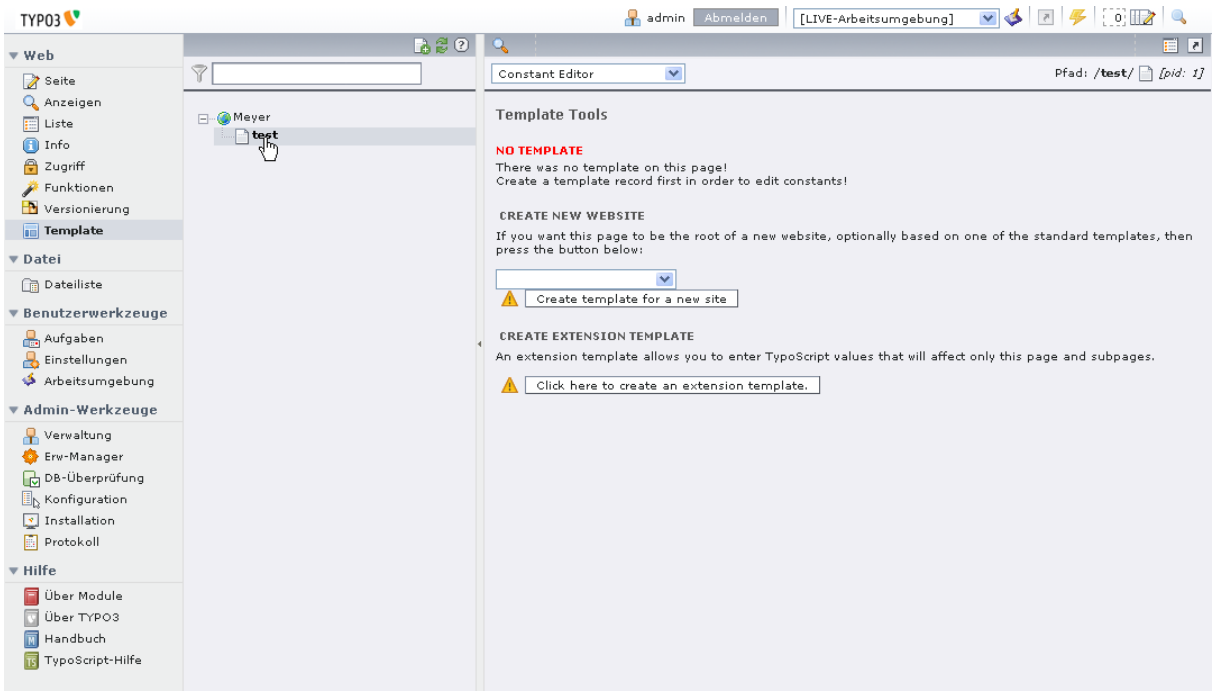
Grundsätzliches zu Templates:

- Ein Template ist ein Datensatz, der auf einer Seite liegt.
- Auf einer Seite können auch mehrere Templates liegen.

- Sämtliche Einstellungen wie auch der TypoScript-Code selber werden in der Datenbank gespeichert.

3.1 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 2) 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 die Schaltfläche "***Create template for a new site***". Im rechten Frame erscheint folgende Seite:

3.1.1 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. Dahinter verbergen sich unterschiedliche Designs und TypoScript-Definitionen, die angepasst werden können. Die Anpassung erfolgt mittels des "Constant Editor" oder direkt über TypoScript.

Hinweis:
In der Regel sollen Internet-Projekte ein individuelles Layout erhalten. Es ist daher nicht ratsam, auf diese vorgefertigten Designs zurückzugreifen. Diese sind sehr veraltet und für den produktiven Einsatz nicht zu empfehlen. 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 "Projekt- oder Root-Template" bezeichnet.

3.1.2 Create an extension template

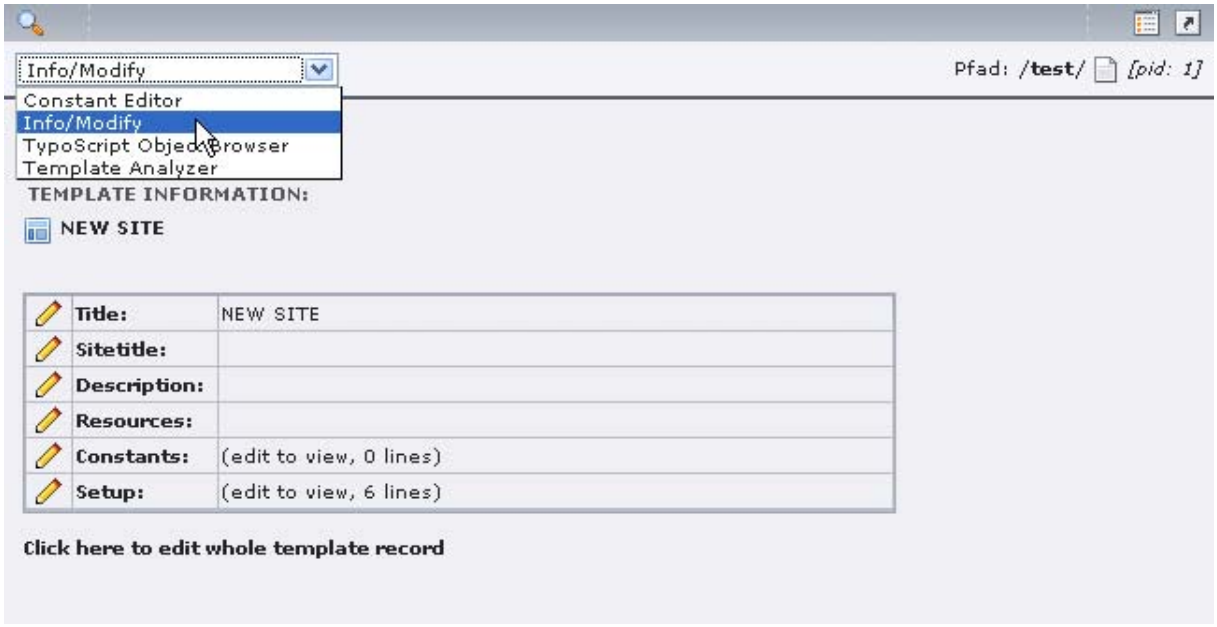
Extension-Templates bieten die Möglichkeit, vererbte Eigenschaften, Objekte, Variablen etc. zu überschreiben. Ein Template wird immer "nach unten hin" vererbt – mit einem Extension-Template wird diese Vererbung nicht unterbrochen, es können jedoch einzelne Ausnahmen definiert werden, die ebenfalls wieder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

Extension-Templates sollten dann angewendet werden, wenn ein Projekt-Template existiert (siehe 3.1.1), es also eine generelle Definition der Internetseite gibt, die aber ab einer bestimmten Seite anders sein soll.

3.2 Info/Modify

Im oberen Teil sehen wir eine Auswahlbox, die mehrere Elemente besitzt. Eines dieser Elemente ist "Info/Modify", mit dem wir überwiegend arbeiten werden.

"Info/Modify" bietet die Möglichkeit, einzelne Bereiche eines Templates gezielt zu bearbeiten. Hier wird das TypoScript der Präsentation im Feld „Setup“ beschrieben. Wenn "Info/Modify" ausgewählt wird, steht ein "Kasten" mit mehreren Feldern zur Verfügung:



3.2.1 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.

3.2.2 Sitetitle

Im Feld "***Sitetitle***" kann der Prefix für einen Seitentitel angegeben werden. Wenn eine Seite im Frontend betrachtet wird, wird in der Regel nur der Titel der aktuellen Seite als HTML-Title-Tag aufgenommen. Soll z.B. der Firmenname immer miterscheinen, so kann dieser Firmenname im Feld "Sitetitle" angegeben werden. Der erzeugte HTML-title-Tag wäre dann "Sitetitle:title", also z.B. "Mustermann AG : Homepage".

3.2.3 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.

3.2.4 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.

3.2.5 Constants

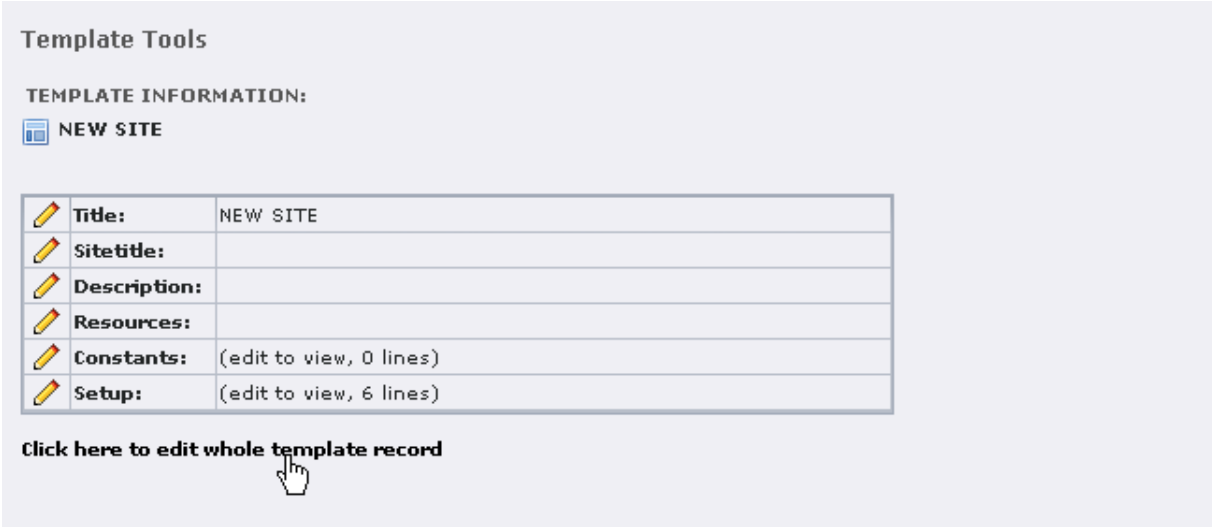
Unter "***Constants***" können Variablen definiert werden, die in TypoScript als Constanten ausgelesen werden. Das Feld "Constants" enthält kein TypoScript!

3.2.6 Setup

Im Feld "***Setup***" wird TypoScript-Code verwendet.

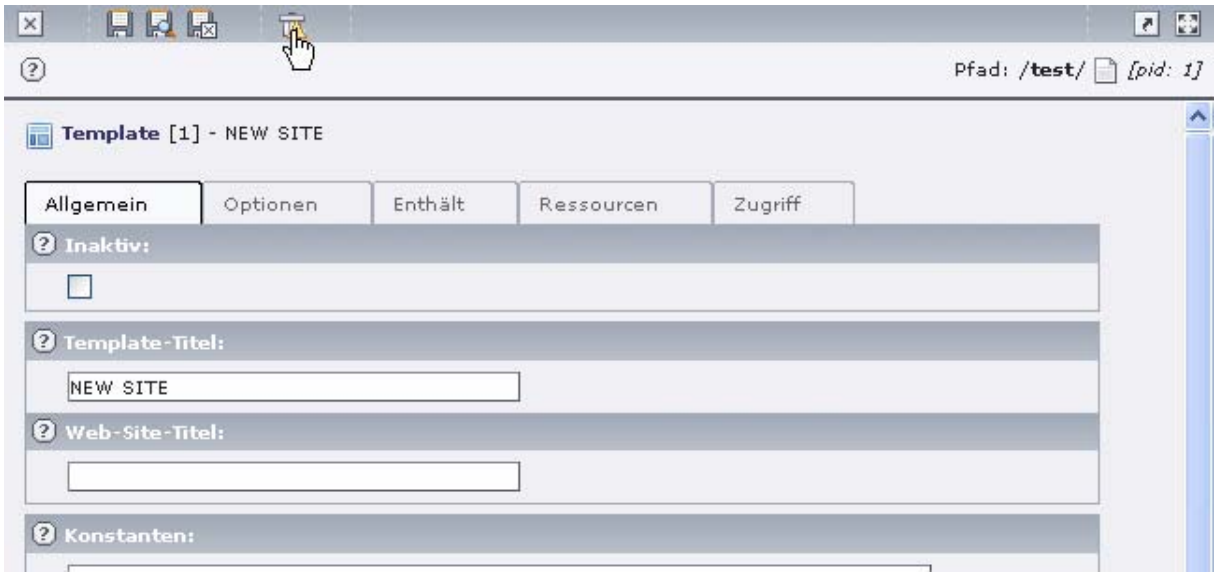
3.3 "whole Template"-Record

Unterhalb der beschriebenen Felder ist ein Textlink vorhanden „***Click here to edit whole template record***“. Hier kann in das gesamte Template eingegriffen werden. Es stehen sowohl die Felder aus dem Kapitel 3.2 zur Verfügung als auch noch einige weitere, wie z.B. "***Include Static***".



3.3.1 Template löschen/inaktiv setzen

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 "Inaktiv" auf dem Reiter „Allgemein“ 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.

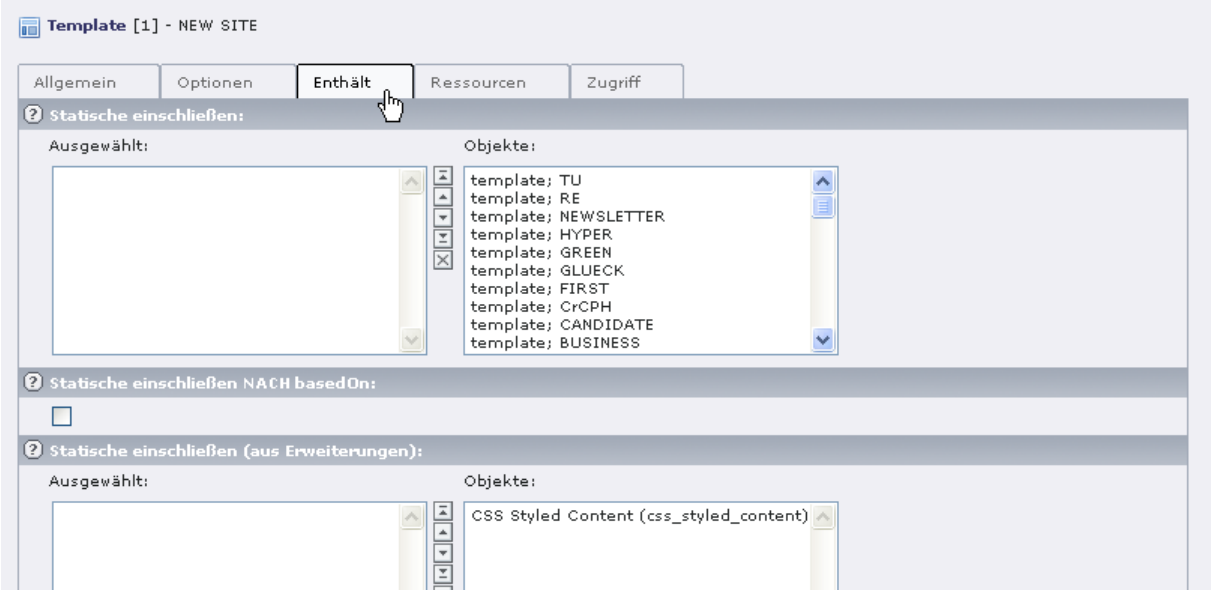
3.3.2 Clear Constants/Setup

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

3.3.3 Include static

Auf dem Reiter „***Enthält***“ können über die Felder „**Statische einschließen“** so genannte „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

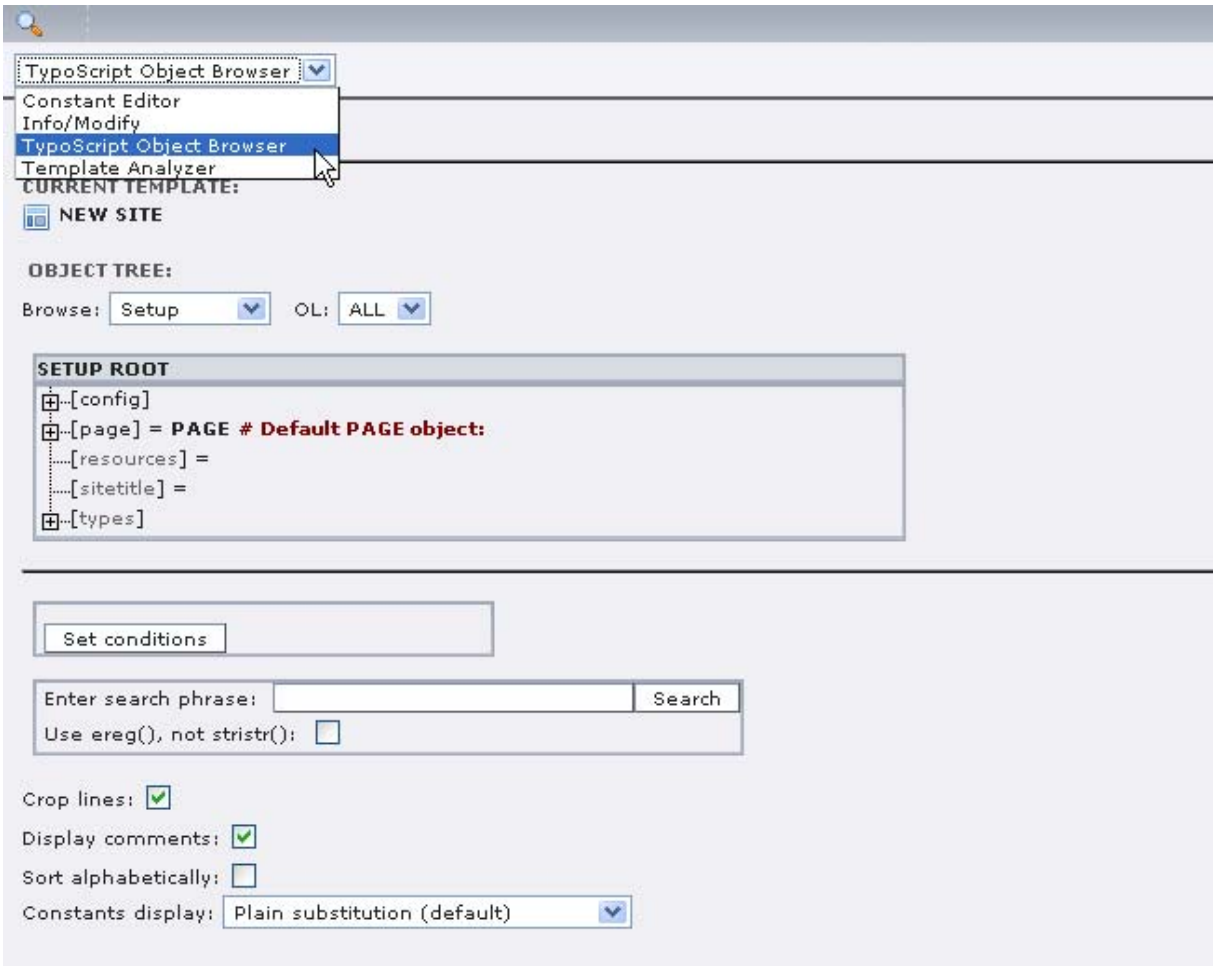
Templates zur Verfügung stehen, können Sie im Install-Tool unter „Database-Analyser" -> „Import" diese statischen Templates zur Verfügung stellen.



Wir werden an späterer Stelle (Kapitel 5.4) das statische Template "css_styled_content" inkludieren. Dieses statische Template enthält eine funktionsfähige und ausgearbeitete Definition, wie Inhalte dargestellt werden können.

3.4 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 aufbereiteten" TypoScript-Code in Form eines Baumes an. Es werden Fehler angezeigt und eine Suche steht zur Verfügung.



3.5 Template Analyser

Mit dem Template-Analyser können komplexere Template-Strukturen "analysiert" werden. So gibt der Template-Analyser zum Beispiel Aufschluss darüber, in welcher Reihenfolge Templates 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.

3.6 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 3.2.5) 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.

3.7 Das TYPO3 Caching-Konzept

TYPO3 ist dynamisch. Alle Inhalte werden somit in Datenbanken gehalten und „auf Abruf“ entsprechend erzeugt. Um die Serverlast bzw. Performance zu erhöhen, wurde ein Caching eingeführt. Hierdurch werden sowohl Inhalte als auch vom System erzeugte Grafiken statisch gehalten, bis sich Veränderungen ergeben.

Die Datenbankabfragen zum Erzeugen einer Seite sind so komplex, dass Sie für große Präsentationen mit vielen Zugriffen nicht ständig ausgeführt werden können. Es sind in der Regel auch die gleichen Daten, die wiederkehrend vom Webserver ausgeliefert werden, solange dynamischen am Inhalt keine Änderungen durchgeführt werden.

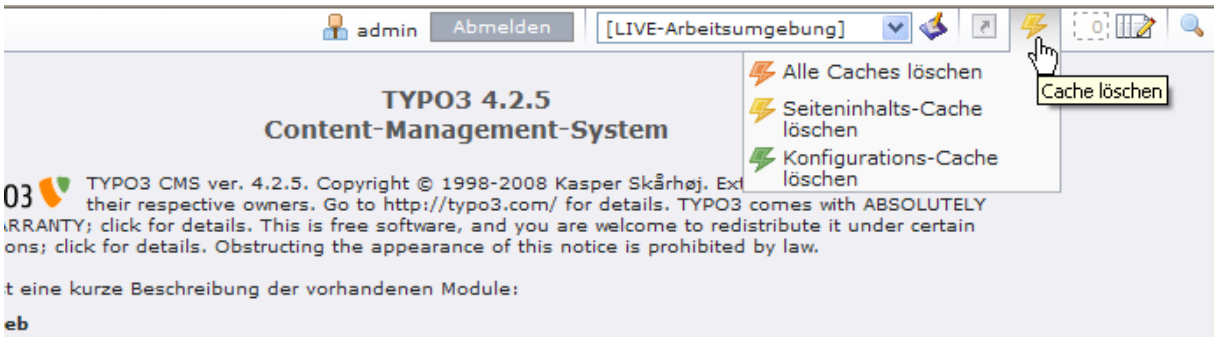
Hierzu speichert TYPO3 seine „Ergebnisse“ einer Datenbank- Abfrage in einer Caching-Tabelle – ebenfalls in der Datenbank. Komplexe Abfragen über mehrere Tabellen sind somit nicht mehr notwendig, da TYPO3 einmal gespeicherte Ergebnisse aus einer einzelnen Tabelle holen kann.

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, dass 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 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.

Das Leeren des Caches ist über die Schaltfläche „cache löschen“ (gelber Blitz) im oberen Teil des TYPO3 Backends möglich.



Das Löschen des Seiteninhalts-Caches (FE-Cache) bewirkt ein Leeren (MYSQL Truncate) der cache_* Tabellen in der Datenbank. Der Konfigurations-Cache (BE Cache) löscht die temp_* Dateien im Verzeichnis typo3conf/.

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 Sie dynamische Grafiken erstellen, erfahren Sie in den Kapiteln 4.11 (GIFBUILDER) und 4.14 (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 4.7 sowie 5.2.

4. TypoScript Grundlagen

4.1 TypoScript-Syntax

Um das Grundverständnis von TypoScript besser zu vermitteln, lösen wir uns 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.

4.1.1 Losgelöst von TypoScript

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 Script-Entwickler 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.

In TypoScript können verschiedene Operatoren verwendet werden: Zuweisen, modifizieren. kopieren, referenzieren 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-, Referenz- 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.

4.1.2 Operatoren

Operator	Beschreibung

=	**Wertzuweisung**
	Bsp:
	`seite = PAGE`
	`seite.10 = TEXT`
	`seite.10.value = HELLO WORLD`

>	**Lösch-Operator**
	Der Lösch-Operator löscht alle 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.10 = TEXT`
	`seite.10.value = HELLO WORLD`
	`seite.20 < seite.10`

=<	**Referenz-Operator**
	Wird ein Objekt als Referenz eines anderen Objektes angelegt und die Eigenschaft verändert, so ändern sich die Eigenschaften bei beiden Objekten
	Bsp:
	`seite = PAGE`
	`seite.10 = TEXT`
	`seite.20 =< seite.10`
	`seite.10.value = HELLO WORLD`

:=	**Modifikation**
	Seit TYPO3 Version 4.0 können Wert zu Laufzeit geändert werden.
	Bsp.:
	`seite = PAGE`
	`seite.10 = TEXT`
	`seite.10.value = HELLO WORLD`
	`seite.10.value := appendString(und TYPO3)`
	`seite.10.value = HELLO,WORLD,TYPO3`
	`seite.10.value := removeFromList(WORLD)`

{ }	Die geschweiften Klammern dienen zur vereinfachten Schreibweise (kein wirklicher Operator).
	Bsp.:
	`seite = PAGE`
	`seite {`
	`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>`
	`)`

#
//
/* */
	Kommentare
	Bsp.:
	`# Das ist ein Kommentar`
	`// Das ist auch ein Kommentar`
	`/* Das`
	`Ist ein`
	`Mehrzeiliger Kommentar`
	`*/`

4.2 PAGE-Objekt

Das PAGE-Objekt ist ein Top-Level-Objekt, welches in TypoScript zur Verfügung gestellt wird und die Konfiguration und Darstellung einer Seite steuert.

Beispiel 1:

```
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 TLO Objekt „***PAGE***“ zugewiesen. Damit hat „seite“ die PAGE-Eigenschaften erhalten (siehe PAGE-Referenz).
- In einem Template sollte es immer ein PAGE-Objekt mit einer typeNum = 0 geben. Der typeNum wird weiter unten aufgrund seiner Bedeutung ein eigener Abschnitt (4.2.1) gewidmet.
- In Zeile 3 wird der Eigenschaft bodyTag (Case-Sensitiv: bodyTag wird mit einem großen „T“ geschrieben!) der gesamte HTML-Tag angegeben.

In dem Beispiel in dieser Dokumentation wir als Bezeichner ***seite*** verwendet. Sie können hier einen beliebigen Bezeichner wählen, daher haben wir absichtlich nicht das Wort *page* als Bezeichner verwendet. Einige Extensions gehen jedoch von dem Bezeichner *page* aus, daher sollten Sie page = PAGE im Life-Betrieb nutzen.

Beispiel 2:

```
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“.

4.2.1 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). Auch kann die typeNum für diverse Ausgabemedien (Browser, PDA etc.) verwendet werden.

4.2.2 Wie wird die Eigenschaft 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 7.3)

4.2.3 Die Nummern „10, 20, 30, ...“

An einigen Stellen in TypoScript wird man diese Nummern wieder finden. 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 4.5). 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 OBJEKT4 ausgeführt (Position 7), dann OBJEKT2 (Position 20), dann OBJEKT1 (Position 35), dann OBJEKT3 (Position 7).

4.3 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 4.1 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 statischen 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 4.3 vorgestellt.

4.4 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:

4.4.1 stdWrap

Einfache Dynamik kann z.B. mit der Funktion „field“ erreicht werden. Field liest 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 so lange verwendet, bis wir TYPO3 mitteilen, dass es mit einer anderen Tabelle arbeiten soll. Im Kapitel 4.9 (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 aktuellen Seite ausgegeben.

4.4.2 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 erzielt 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.

4.5 COA (Content Objekt Array)

Mit dem Objekt COA kann, ähnlich wie schon im Kapitel 4.1 (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 4.1.

In der Praxis wird dieses Objekt sehr häufig eingesetzt. Anwendungsfälle folgen im Praxis-Kapitel 5.

4.6 CASE

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

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.

4.6.1 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 4.5 vorgestellten Funktionen angewendet werden, so z.B. die Funktion „.field“, das in der Praxis sehr häufig eingesetzt wird. Aber auch andere Funktionen, wie z.B. key.data = DB:pages:1:title, sind natürlich anwendbar.

4.6.2 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, das wiederum 2 TEXT-Objekte enthält (Zeile 9 + 11).
- Wurde keine Übereinstimmung gefunden, ist also der Inhalt des Datenbankfeldes nicht „test“, wird die default-Eigenschaft in Zeile 14 angesprochen, die ebenfalls 2 TEXT-Objekte enthält.

4.7 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:

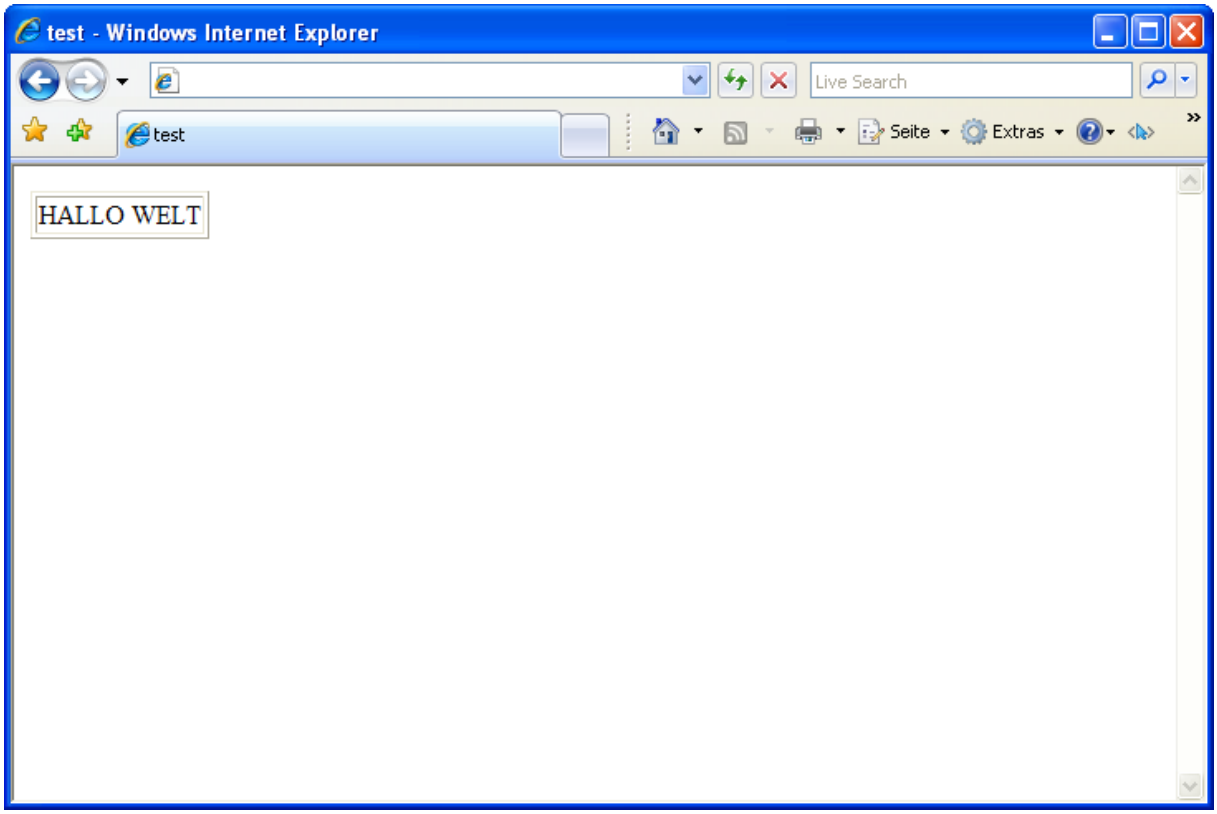
```
<table border="1">
<tr>
<td>
HALLO WELT
</td>
```

```
</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 sichtbar 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 4.8).

4.8 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 4.7 erstellt wurde.

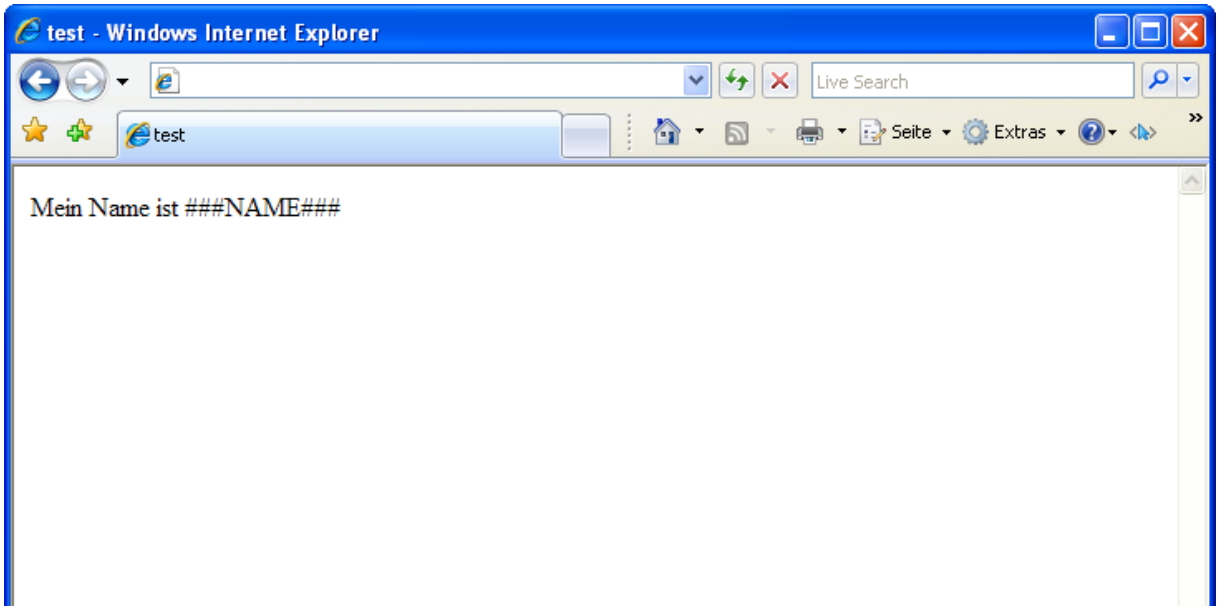
```
01 seite = PAGE
02 seite.typeNum = 0
03
04 seite.10 = TEMPLATE
05 seite.10 {
06     template = FILE
07     template.file = fileadmin/test.txt
08 }
```

Das Frontend-Ergebnis ist identisch mit dem Beispiel von Kapitel 4.7. In TypoScript können wir jedoch erkennen, dass ein Objekt „TEMPLATE“ zwischengeschaltet wurde. Dieses Objekt „TEMPLATE“ hat die Möglichkeit, Platzhalter zu ersetzen.

4.8.1 marks: Platzhalter verwenden

Beispiel 2

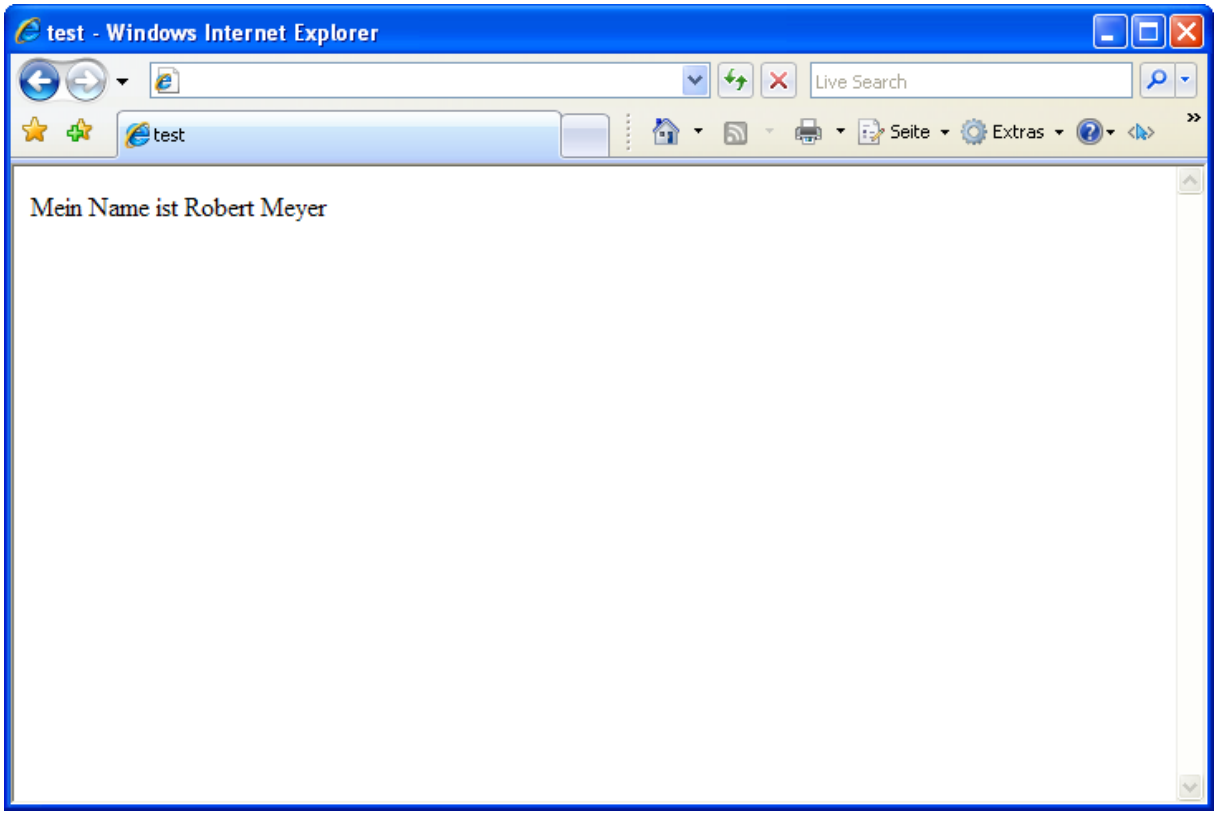
```
01 seite = PAGE
02 seite.typeNum = 0
03 seite.10 = TEMPLATE
04 seite.10 {
05     template = TEXT
06     template.value = Mein Name ist ###NAME###
07 }
```



Beispiel 2 gibt anstelle des Inhaltes einer Datei eine direkt in TypoScript angegebene 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 wird immer in Großbuchstaben geschrieben. Dieses ist keine 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!

4.8.2 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.

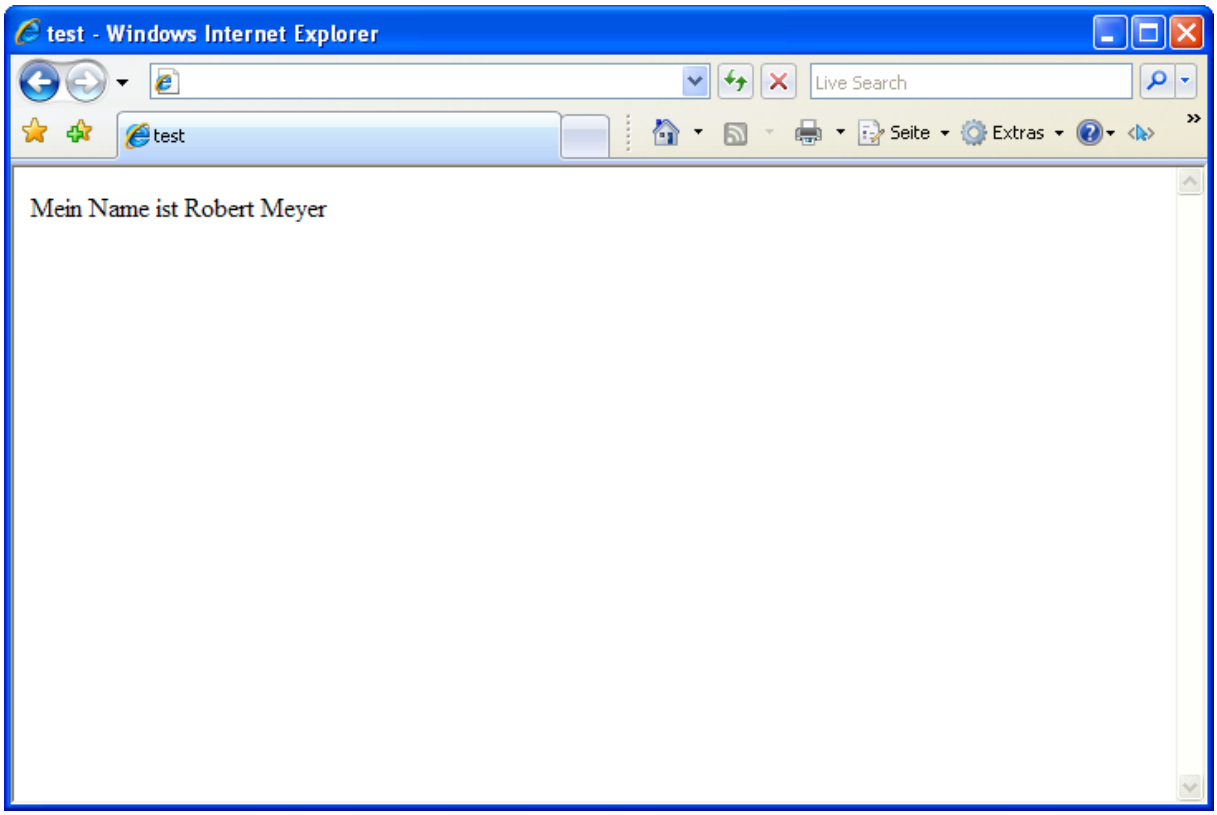
Beispiel 4

Die bestehende Datei „test.txt“, die im Kapitel 4.7 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 }
```



4.8.3 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 }
```

4.9 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.

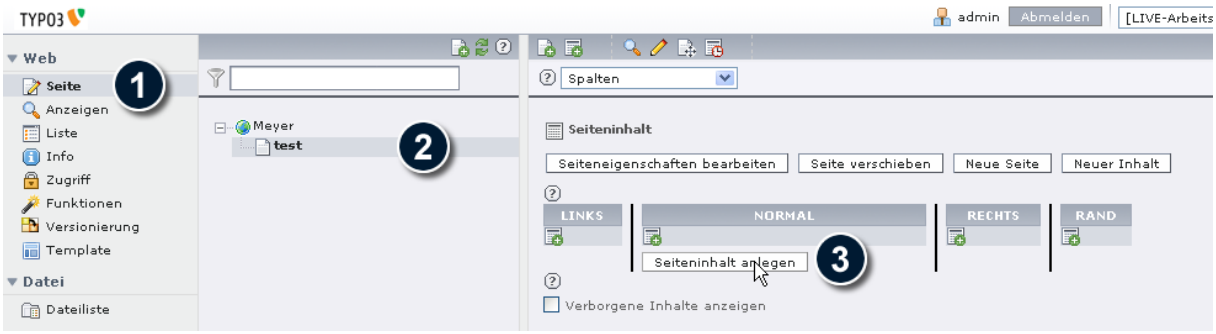
Wird z.B. in TYPO3 ein Seiteninhalt angelegt, so wird dieser Datensatz in der Datenbanktabelle tt_content gespeichert. Jeder Datensatz verfügt über einen Eintrag „pid“ ("parent-ID" = Elternobjekt) und gibt an, auf welcher Seite dieser Seiteninhalt liegen soll. Das Feld „pid“ ist ein Verweis auf die entsprechenden 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.

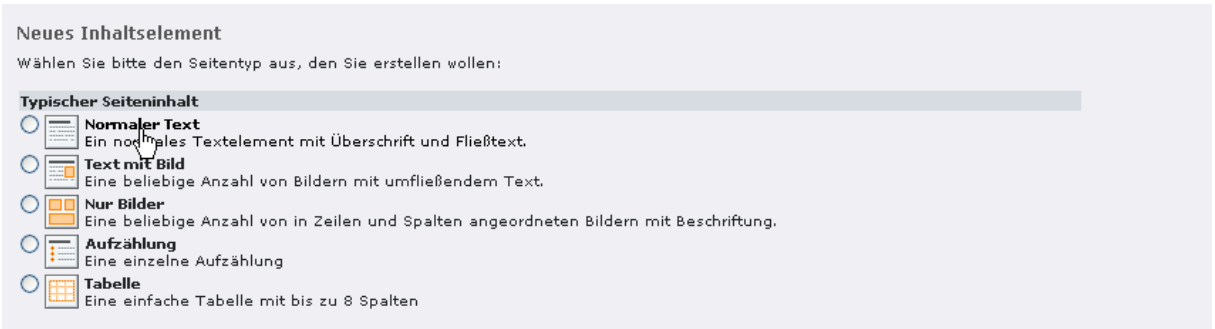
4.9.1 Vorbereitung: Seiteninhalt anlegen

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.

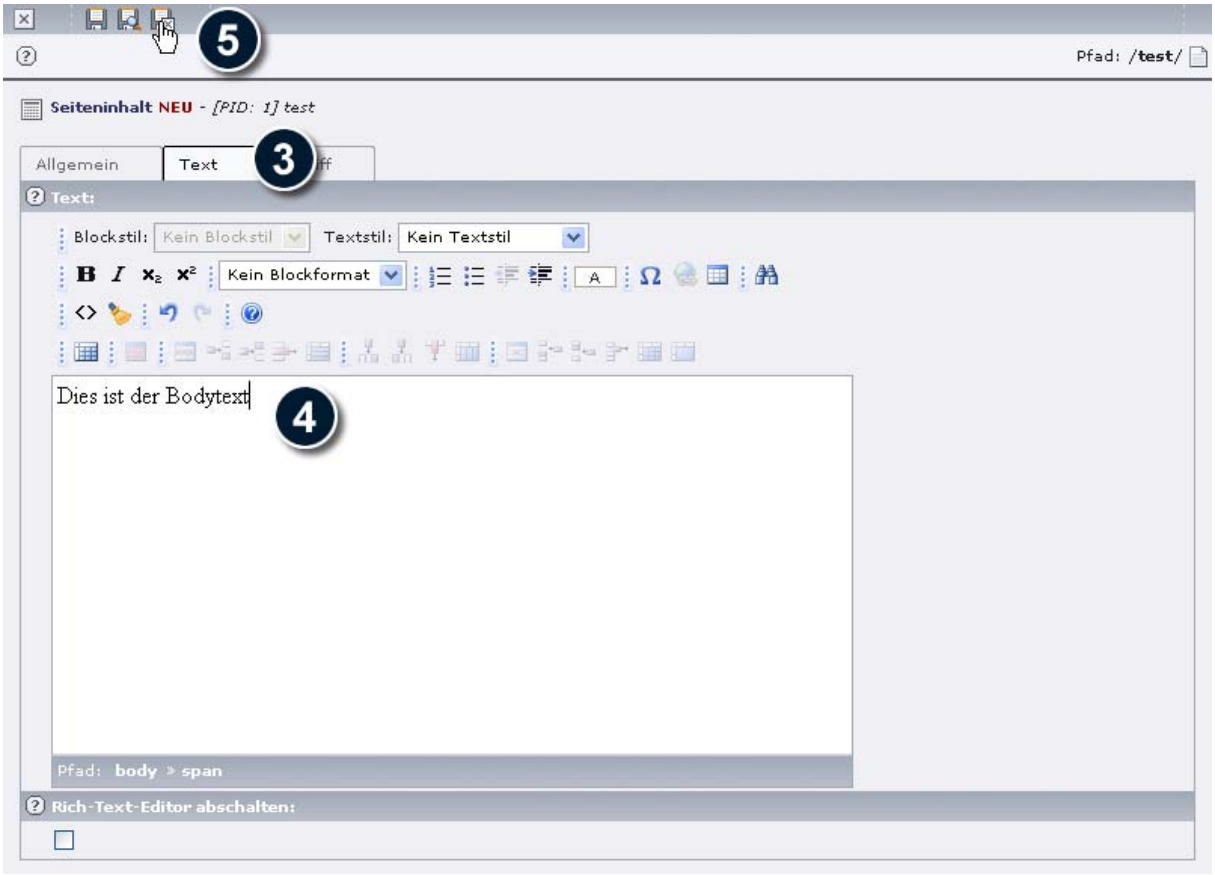


Mit dem Wizard für neue Inhaltselemente bzw. Seiteninhalte können Sie verschiedene vordefinierte Inhaltstypen auswählen. Für unser Beispiel wählen wir den ersten Eintrag „***Normaler Text***“ aus.



Füllen Sie jetzt auf dem Reiter „***Allgemein***“ die Masken „***Überschrift***“ „Dies ist die Überschrift“

Wechseln Sie anschließend auf den Reiter „***Text***“ und füllen Sie das Feld mit dem Inhalt „Dies ist der Bodytext“. Speichern Sie den Seiteninhalt, indem 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 wäre dies aber auch überraschend, da wir TYPO3 noch nicht gesagt haben, dass überhaupt Inhalt angezeigt werden soll. Weitere Informationen sowie eine andere Herangehensweise zum Anlegen von Seiteninhalten finden Sie im Kapitel 5.4.13.

4.9.2 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 als defaultmäßig in der Tabelle „tt_content“ gespeichert.

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 3.3.3 sowie 5.4.13. Die folgenden Abschnitte zeigen den theoretischen Hintergrund auf.

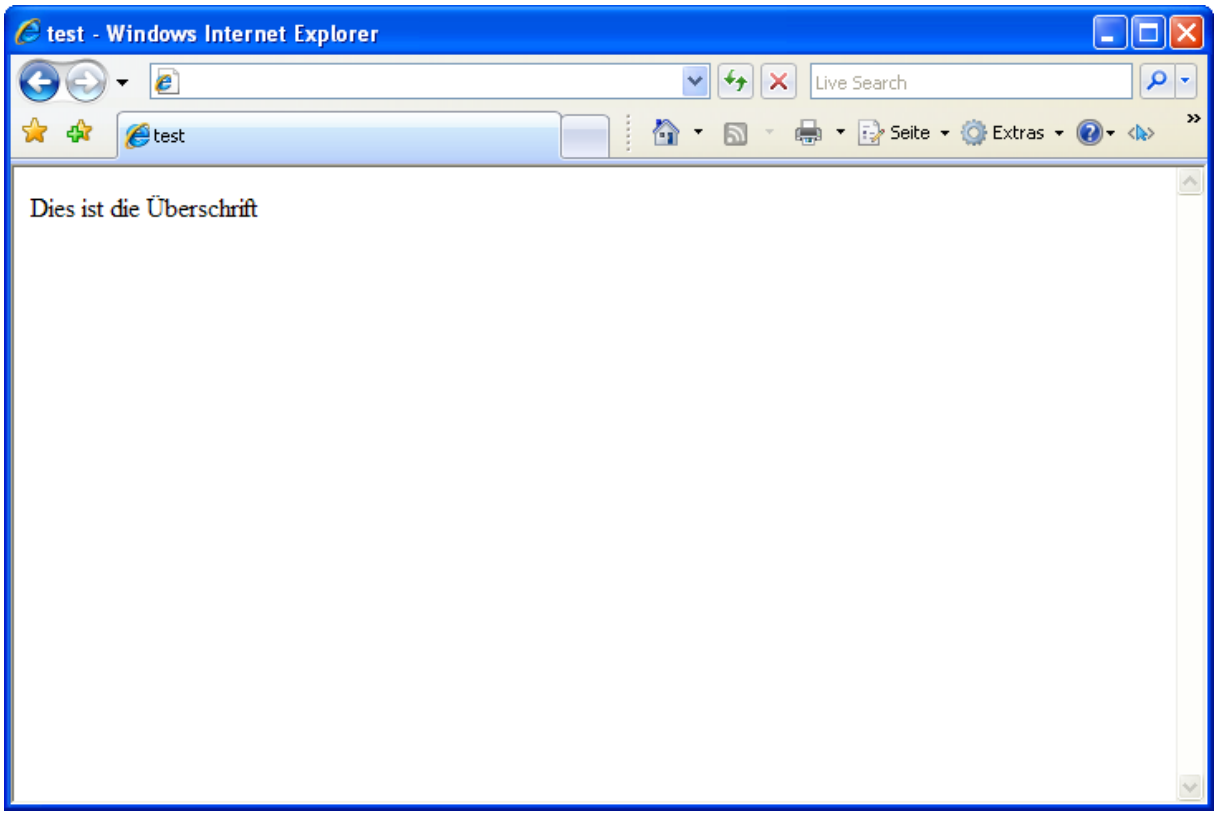
4.9.3 tt_content

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.



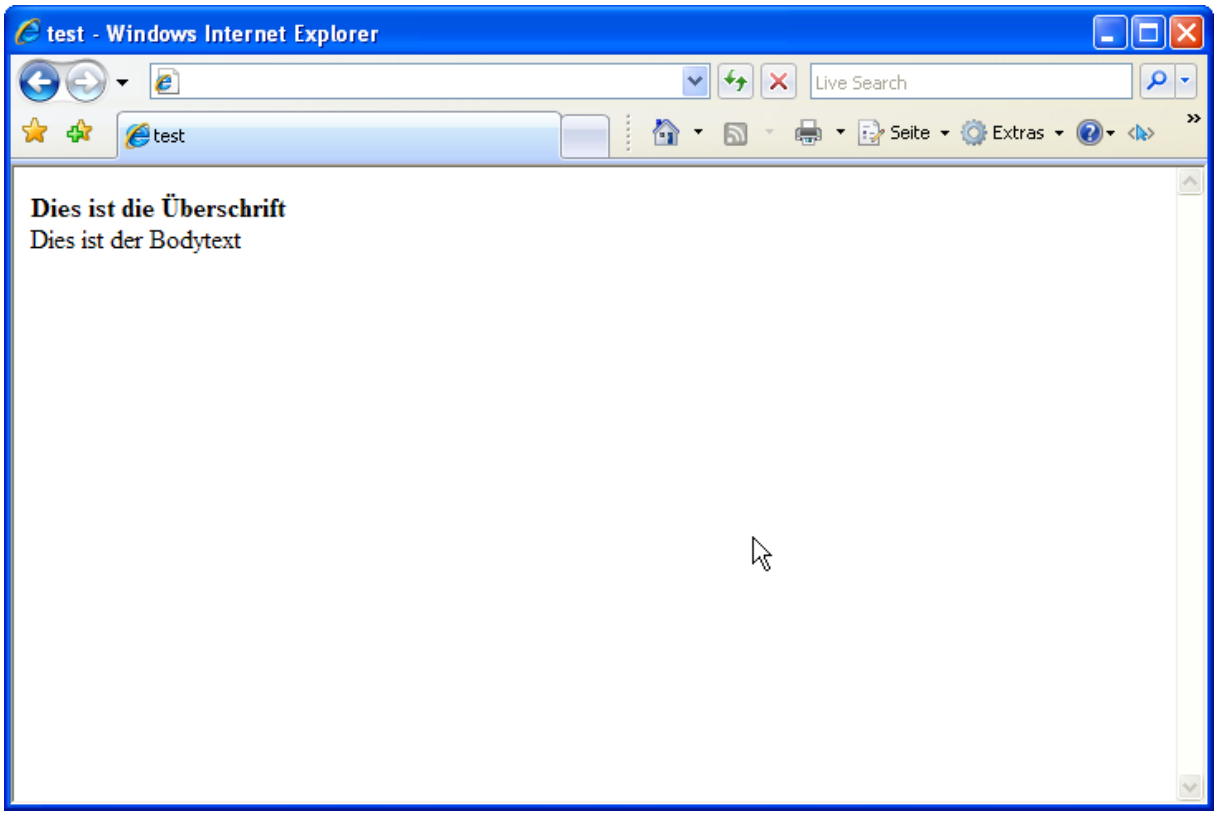
Dies wird uns aber in der Regel nicht genügen. Zur Darstellung sind daher wesentlich komplexere Definitionen notwendig, die hier in Form von einigen Beispielen demonstriert werden sollen.

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 Zeilenumbruch „gewrapt“), an der Position 20 wird der Bodytext ausgegeben.



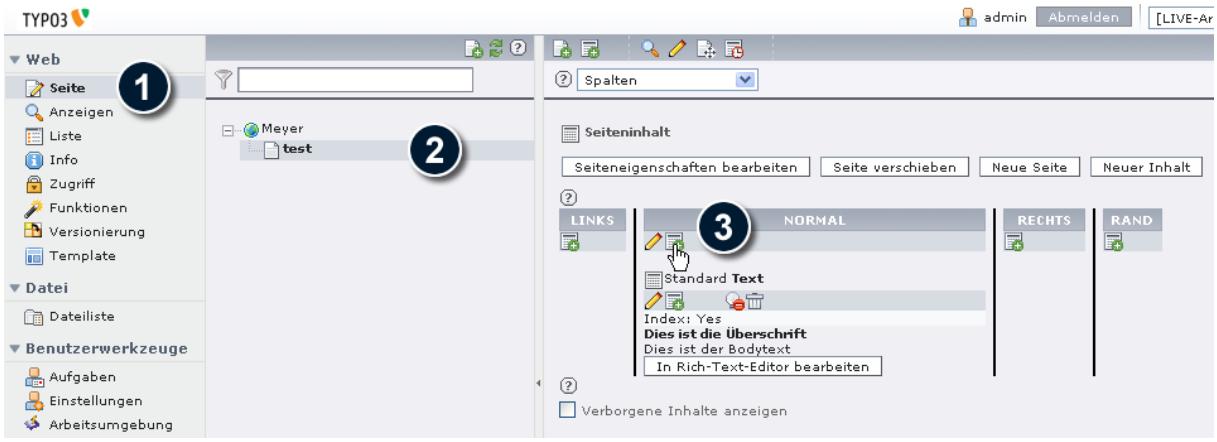
Beispiel 4

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

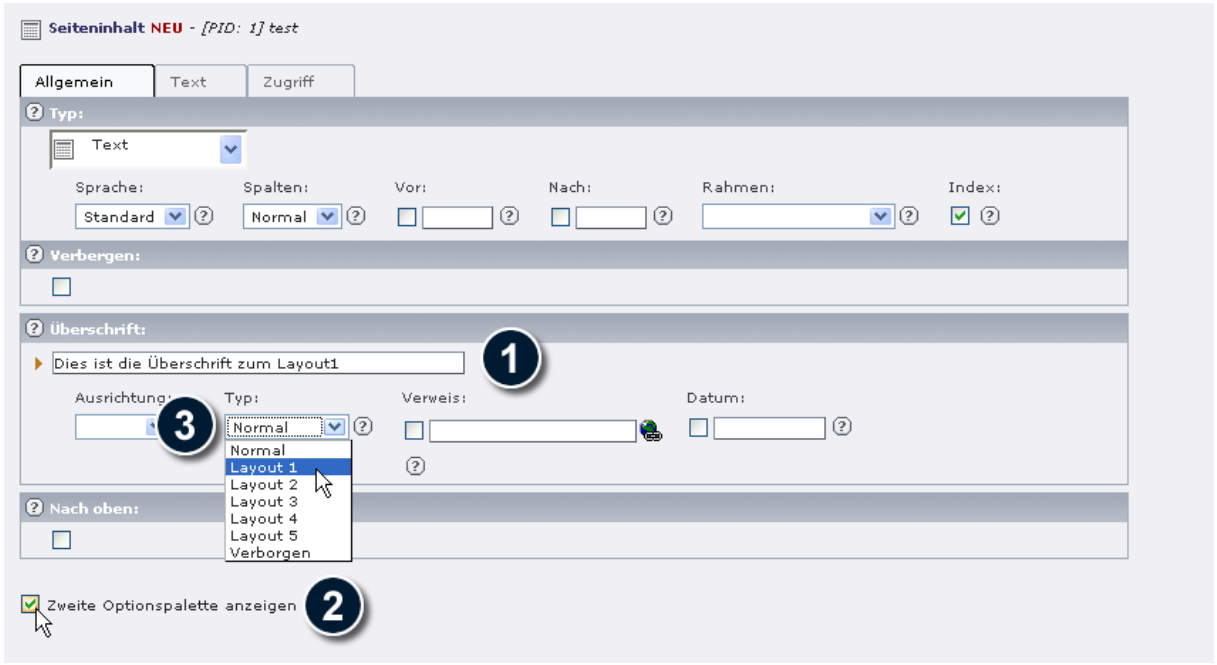
Wir möchten in diesem Beispiel 4 zwei Seiteninhalte mit unterschiedlichen Layouts erstellen und diese auch unterschiedlich darstellen lassen.

Bei dem ersten von uns angelegten Seiteninhalt haben wir das Feld „Typ“ 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.

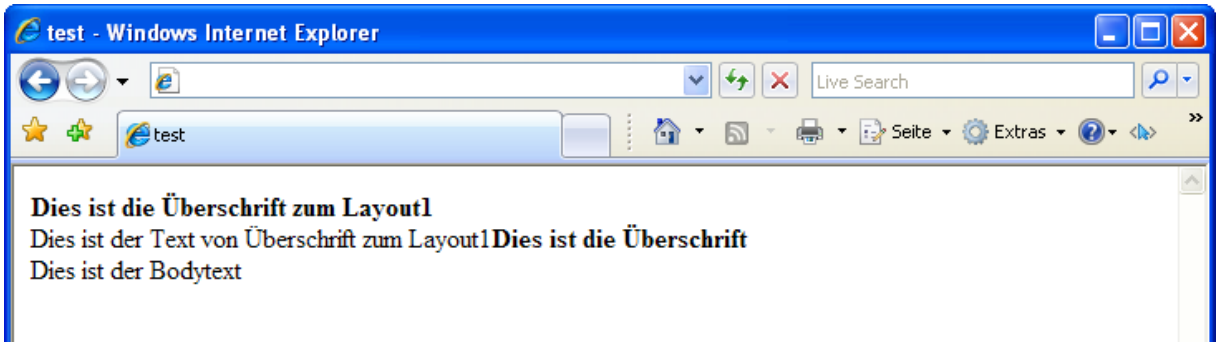
Wir legen nun also einen weiteren Seiteninhalt an – ebenfalls vom Typ „***Normaler Text***“.



Wir füllen das Feld Überschrift mit „Dies ist die Überschrift zum Layout1“, wählen als Typ „***Layout1***“ aus und als Bodytext geben wir eine beliebige Zeichenkette an. Wenn das Feld Typ nicht sichtbar ist, muss die Option „***Zweite Optionspalette anzeigen***“ aktiviert werden.

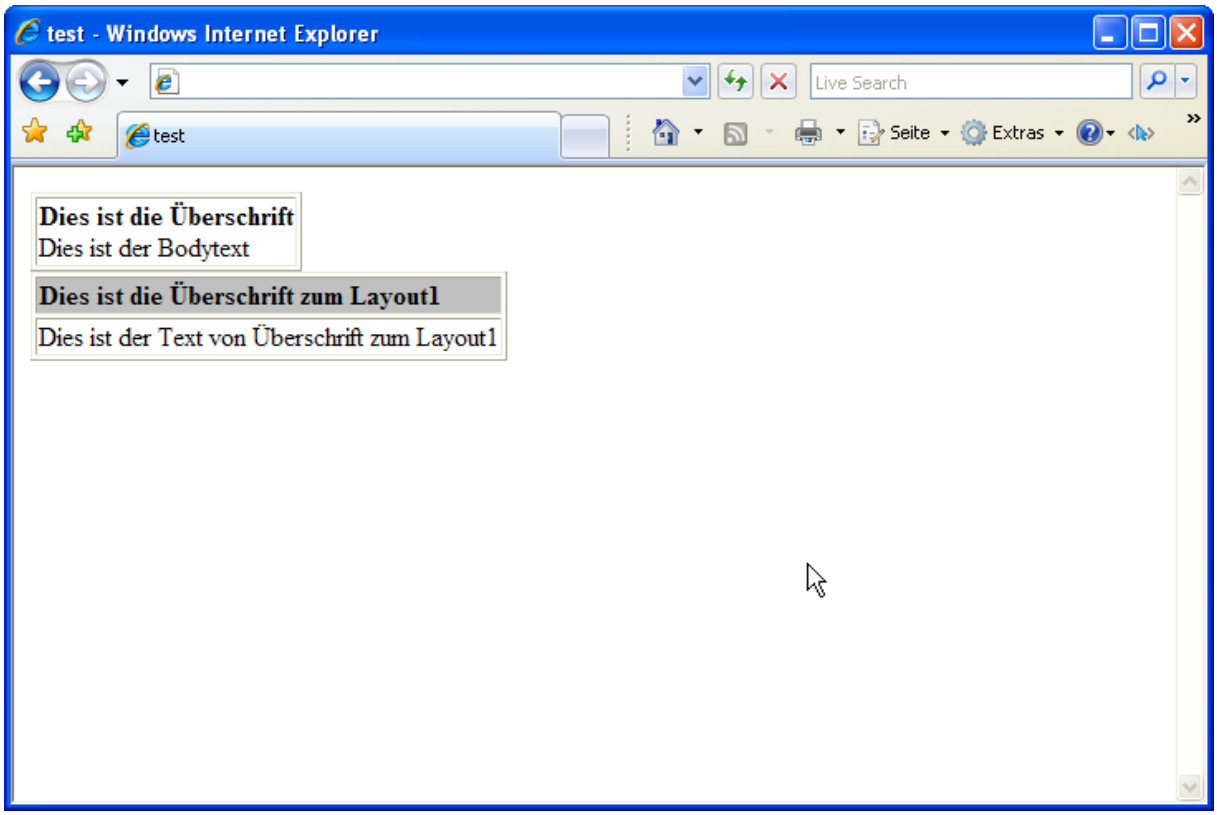


Betrachten wir das jetzige Ergebnis vom Beispiel 3 erneut im Frontend, werden bereits beide Datensätze angezeigt, auch wenn optisch nicht ansprechend.



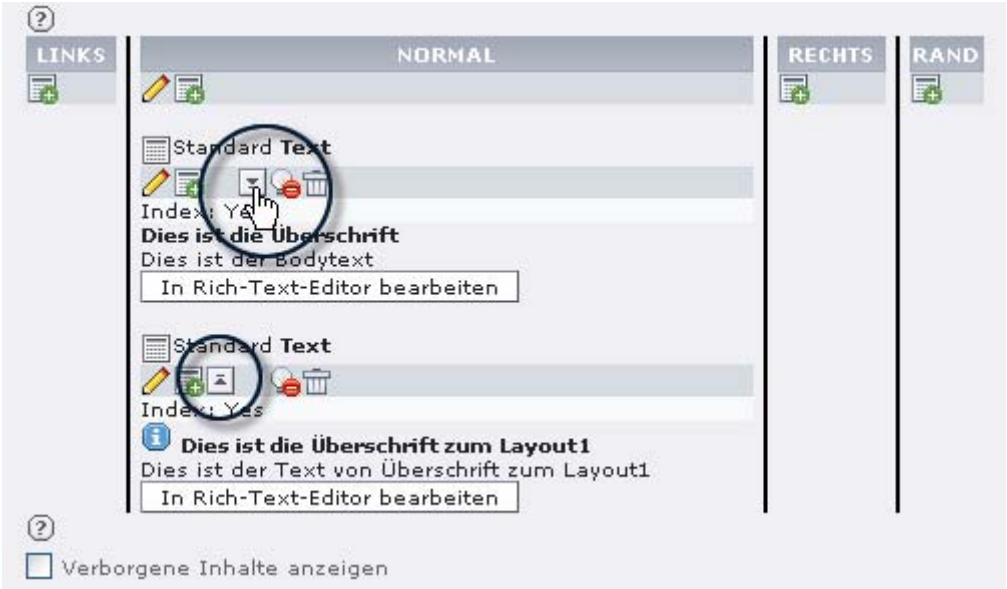
Um das gewünschte Ergebnis mit unterschiedlichen Layouts zu erhalten, müssen wir eine Unterscheidung nach dem ausgewählten Layout machen.

```
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 }
```



4.9.4 select : sortieren

Im TYPO3 Backend können Datensätze in der Reihenfolge verändert werden, um deren Position in einer Spalte zu bestimmen.



Seit der TYPO3 Version 4 wird die Sortierung der Elemente beim Auslesen aus der Tabelle automatisch berücksichtigt. In älteren Versionen würden Sie feststellen, dass die Sortierung nicht funktioniert. Veränderungen der Datensatzreihenfolge im Backend blieben im Frontend ohne Erfolg, daher müssten wirTYPO3 erst mitteilen, wie die Datensätze sortiert werden sollen.

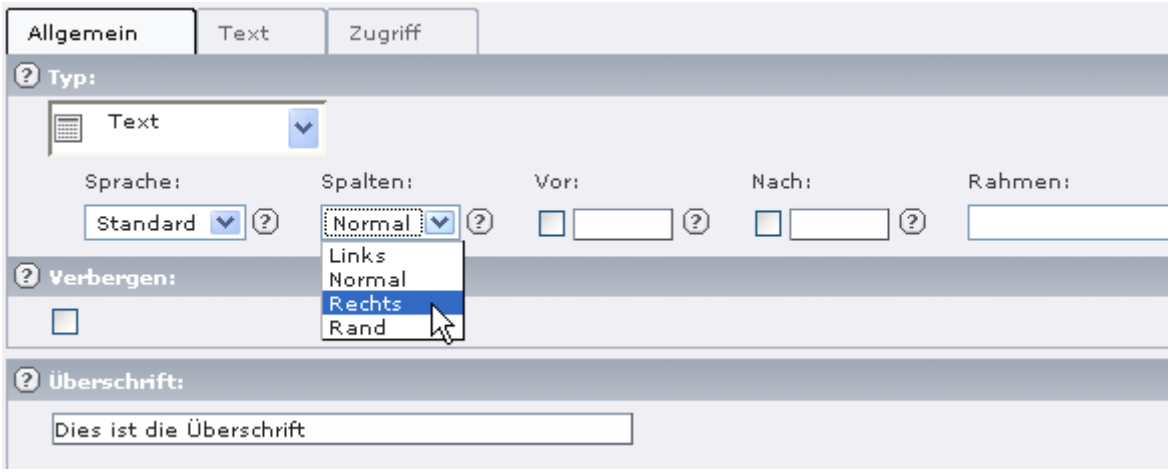
Beispiel 5

Das Beispiel 4 müsste um die Zeile 5 erweitert werden:

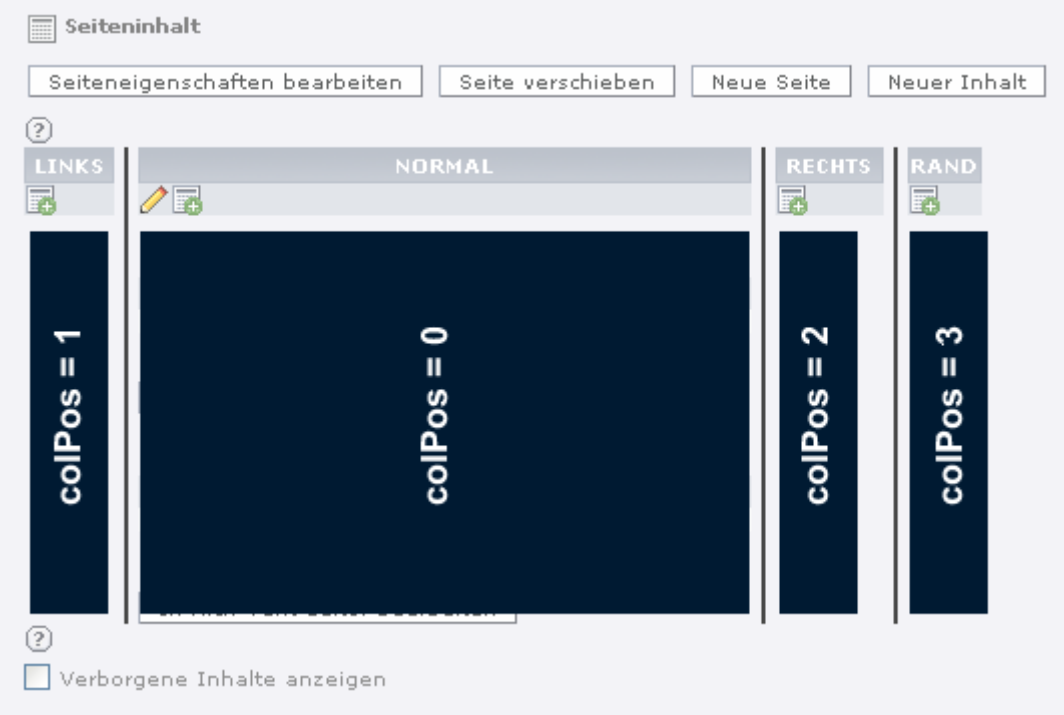
```
01 [...]
03 seite.10 = CONTENT
04 seite.10.table = tt_content
05 seite.10.select.orderBy = sorting
06 [...]
```

4.9.5 select : Spalten

Wer ein Design verwenden möchte, das mit unterschiedlichen Spalten arbeitet, z.B. einem normalen Inhalts-Bereich und einer rechten Spalte, wird im Backend schnell fündig. Hier kann angegeben werden, dass 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, durch das 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 [...]
```

4.10 IMAGE

Mit dem Objekt IMAGE können Grafiken angezeigt werden. TYPO3 übernimmt hierbei die gesamte HTML-Arbeit und erzeugt einen img-Tag.

Beispiel 1

Wir laden zunächst eine Grafik in den Ordner fileadmin/ hoch. Der Dateiname der verwendeten Grafik lautet testbild.jpg. Die Datei speichern wir über das Menü „Dateiliste“ auf den Server in den Ordner fileadmin/. Mittels TypoScript können wir nun die Grafikdatei unter fileadmin/testbild.jpg 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. Es können z.B. Größen des Bildes geändert und 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
```

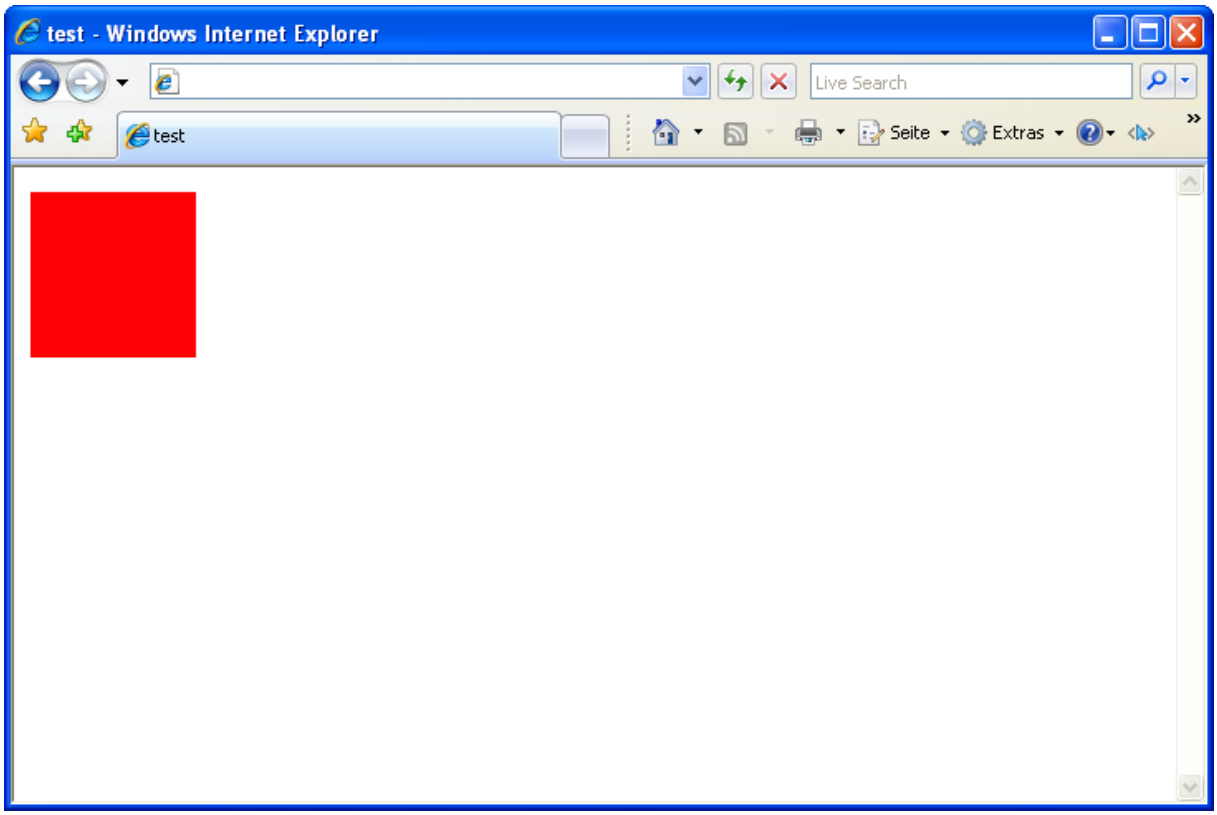
Die gesamte Mächtigkeit der Bildverarbeitungsfunktionen kommt aber insbesondere beim GIFBUILDER-Objekt (Abschnitt 4.11) zum Vorschein.

4.11 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 }
```



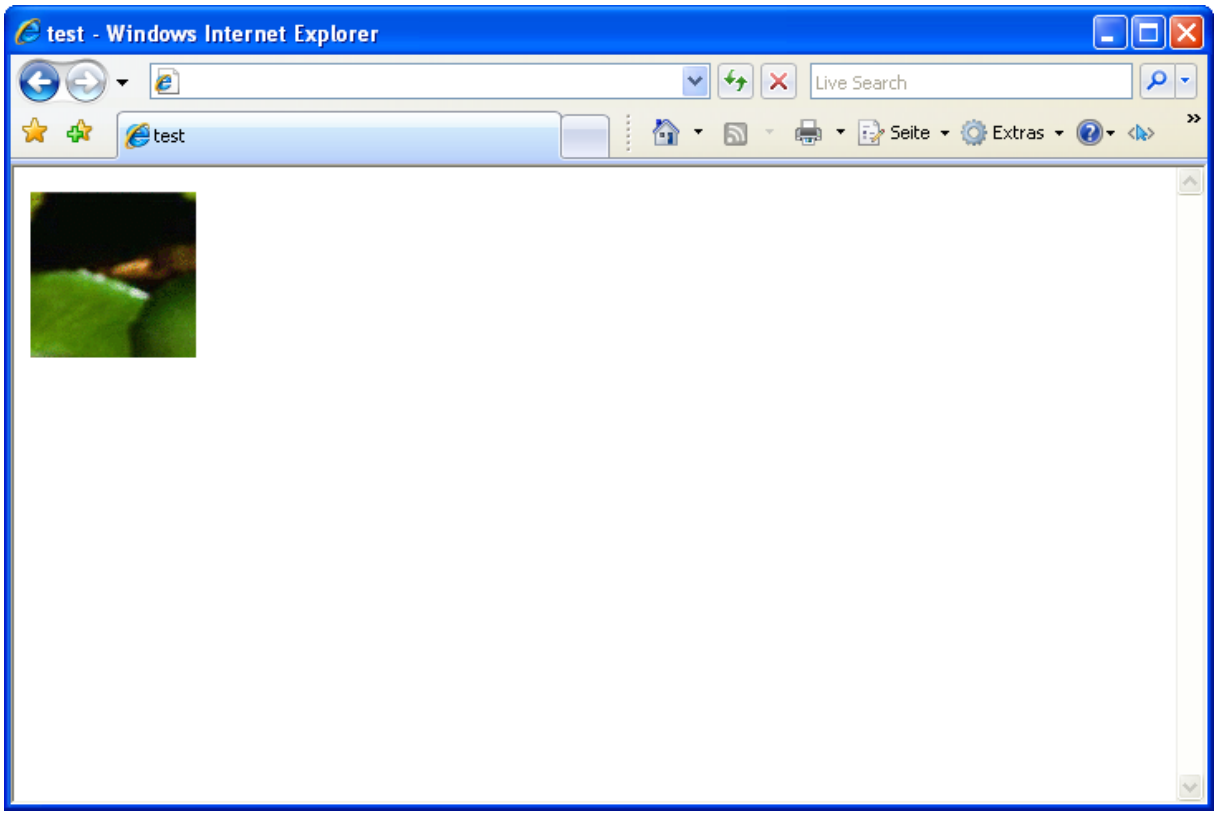
- Im Beispiel wird in Zeile 3 ein IMAGE-Objekt angelegt.
- Eine Grafikdatei ist jedoch noch nicht vorhanden, sondern wird dynamisch erstellt (Zeile 4: "file = GIFBUILDER").
- Für die Grafik wurden 2 Eigenschaften festgelegt: Die Abmaße der Grafik (Zeile 6:100 x 100 Pixel) und die Hintergrundfarbe (Zeile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

4.11.1 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 }
```



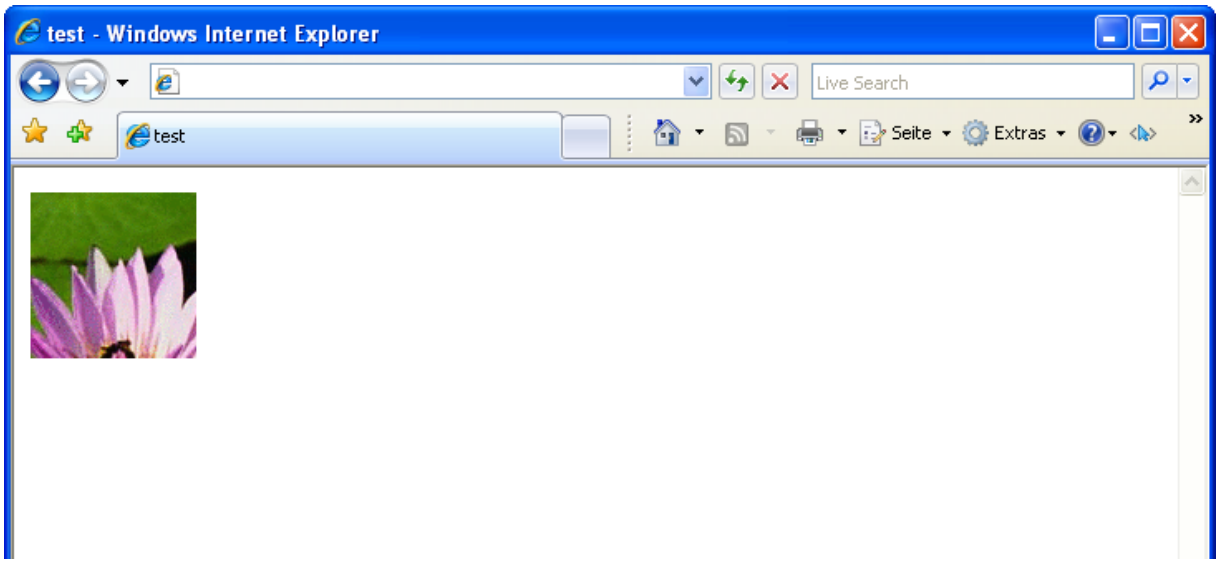
Wir sehen also, dass eine Ebene 10 vorhanden ist (Zeile 8+9). Diese Ebene 10 enthält eine Grafik, die unsere Hintergrundfarbe überdeckt.

4.11.2 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 = -150, -100
11 }
```



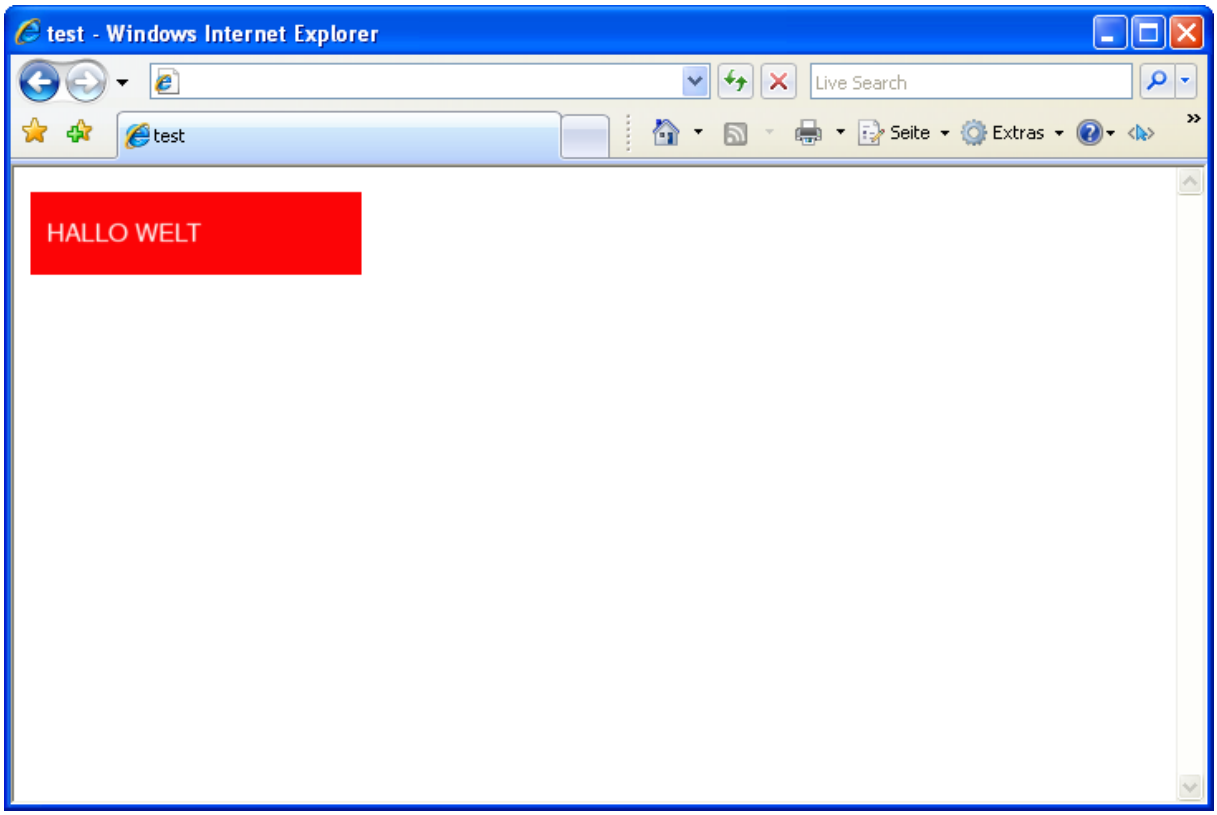
In Zeile 10 wird die Verschiebung der Ebene 10 um 150 Pixel nach links (X Koordinate) sowie 100 Pixel nach oben (Y Koordinate) angegeben.

4.11.3 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 4.3) 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 TEXTObjekt 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 }
```



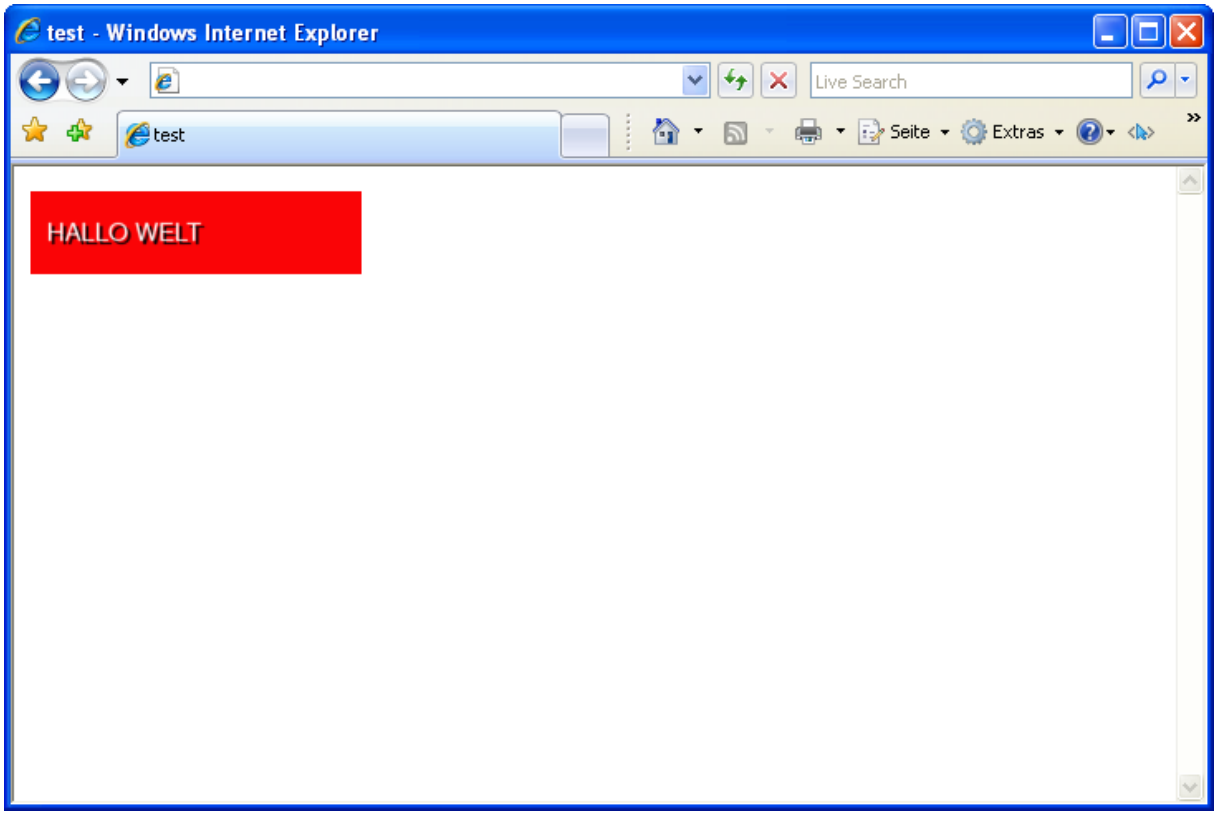
- In Zeile 6 haben wir die Dimensionen der Grafik von 100 x 100 auf 200 x 50 geändert, um einem länglichen Text gerecht zu werden.
- Der Ebene 10 wurde ein grafisches TEXT-Objekt zugewiesen (Zeile 8).
- Die Eigenschaft „text“ in Zeile 9 liefert hier einen statischenText zurück.
- Die Eigenschaften fontFile, 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.

4.11.4 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 }
```



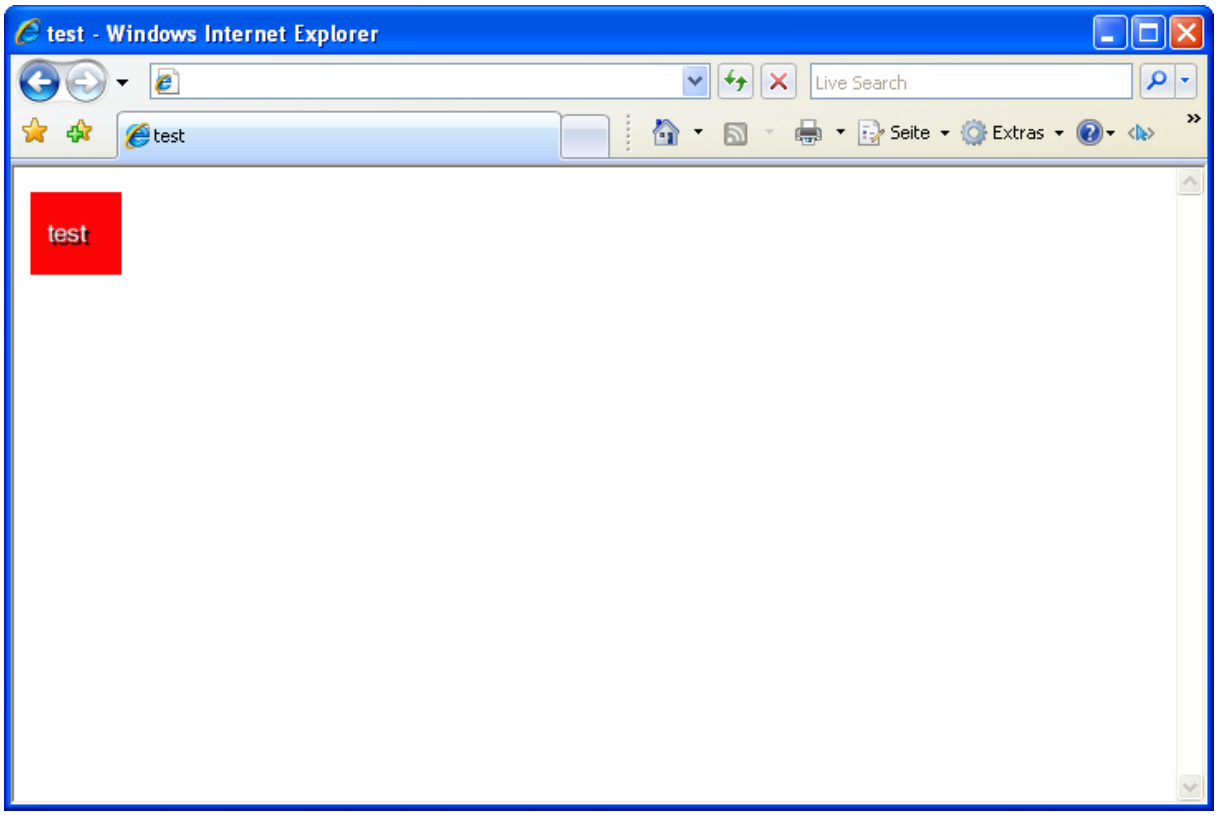
- In Zeile 16 wird die Ebene 10 in die Ebene 5 kopiert. Damit ist die Ebene 5 mit der Ebene 10 absolut identisch und liegt unterhalb von der Ebene 10.
- In den Zeile 17 und 18 nehmen wir entsprechende Modifikationen vor, um einen Schatteneffekt zu erreichen (Schwarzer Text, Positionieren um 2 Pixel weiter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

4.11.5 Mehr Dynamik

Unsere bisher erstellten Grafiken werden zwar dynamisch erstellt, die tatsächliche Dynamik fehlt jedoch. Im folgenden 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 der Text vom Rand 10 Pixel entfernt steht (Zeile 14).

4.12 HMENU

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

4.12.1 Einführung

TYPO3 liefert von Hause aus eine automatische Grafikerzeugung (Kapitel 4.10 und 4.11). Diese 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 beliebige Menüs erstellen, die ein Verhalten und Aussehen nach Ihren Wünschen haben.

4.12.2 Menüarten

Es gibt 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.

4.12.3 Zustände von Menüelementen

Menüelemente (bzw. Menüeinträge) können unterschiedliche Zustände annehmen. Diese sind sehr vielfältig und unterschiedlich kombinierbar. Eine genaue Übersicht der möglichen Zustände können Sie der TSref entnehmen. Die meist genutzten sind:

Beschreibung	Zustand
NO	NO beschreibt den normalen Zustand eines Menüeintrages
RO	RollOver bzw. MouseOver sind Änderungen z.B. an der Schrift, der Grafik, der CSS Formatierung, die bei einem Herüberfahren mit der Maus beschrieben werden.
ACT	Der aktuelle Verlauf. Dies bedeutet, dass 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 Zustände kann beliebig definiert werden. Logischerweise sind die unterschiedlichen Definitionen jedoch sehr gering und dezent: Eine andere Schriftart, ein anderer Hintergrund oder eine andere CSS Klasse 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
```

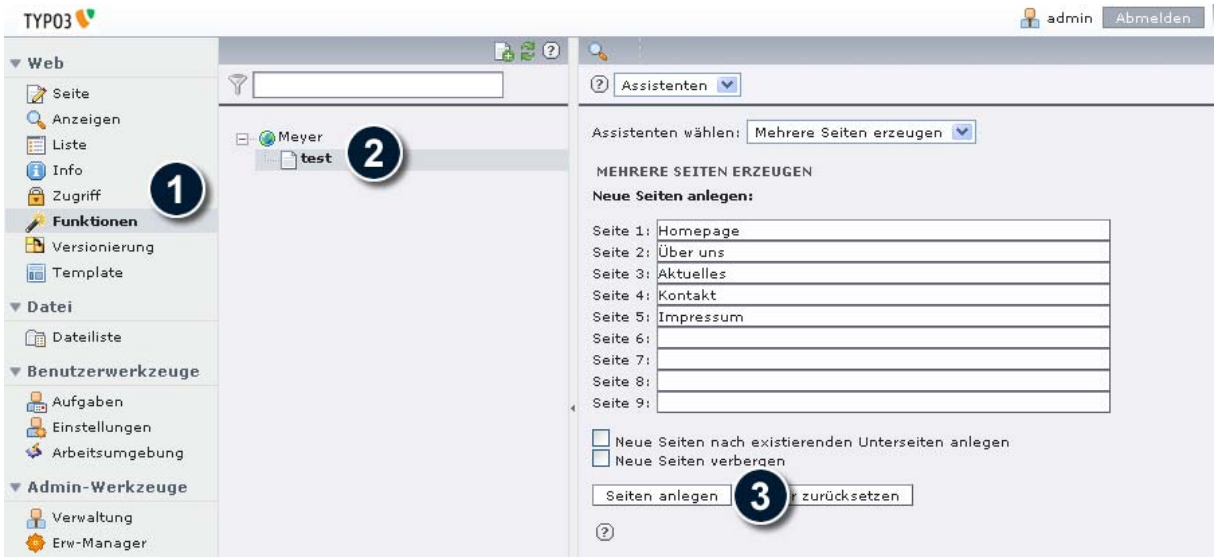
Ebenfalls können die Zustände untereinander kombiniert werden: CURRO, CURIFSUB, ACTIFSUBRO (siehe TSref, Menu Objects).

4.12.4 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***“ und anschließend auf den textlink der Seite „test“ im Seitenbaum. Wir sehen rechts eine Maske mit der Möglichkeit, bis zu 9 Unterseiten schnell anzulegen.

Bevor wir nun aber die 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 Anklicken des Plus-Symbols) sollte nun so aussehen:

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

4.12.5 special – was für ein Menü?

Mit der Eigenschaft „special“ des Content Objects (CObjects) HMENU kann angegeben werden, was für ein Menü erstellt werden soll. Dabei kann die Eigenschaft folgende Werte aufnehmen:

Wert	Beschreibung
Directory	Baut ein Menü ausgehend von einer angegebenen Seite auf (Unterseiten der angegebenen Seite).
list	Angabe von beliebigen Seiten, aus denen ein Menü erstellt werden soll.
Updated	Menü der zuletzt geänderten Seiten. Über Parameter kann z.B. das maximale Alter angegeben werden.
Browse	Es wird ein „Blättermenü“ erzeugt, über das Seitenebenen vor- und zurückgeblättert werden können.
rootline	Klickpfad/Brotkrumenpfad (engl. Breadcrumb): "Sie befinden sich hier:".
keywords	Es wird ein Menü von Seiten erzeugt, die Schlüsselwörter enthalten, die auch auf der aktuellen Seite vorhanden sind.
language	Erzeugt ein Sprachauswahlmenü

4.12.6 special : directory

Häufig wird die Eigenschaft special = directory genutzt. 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
```

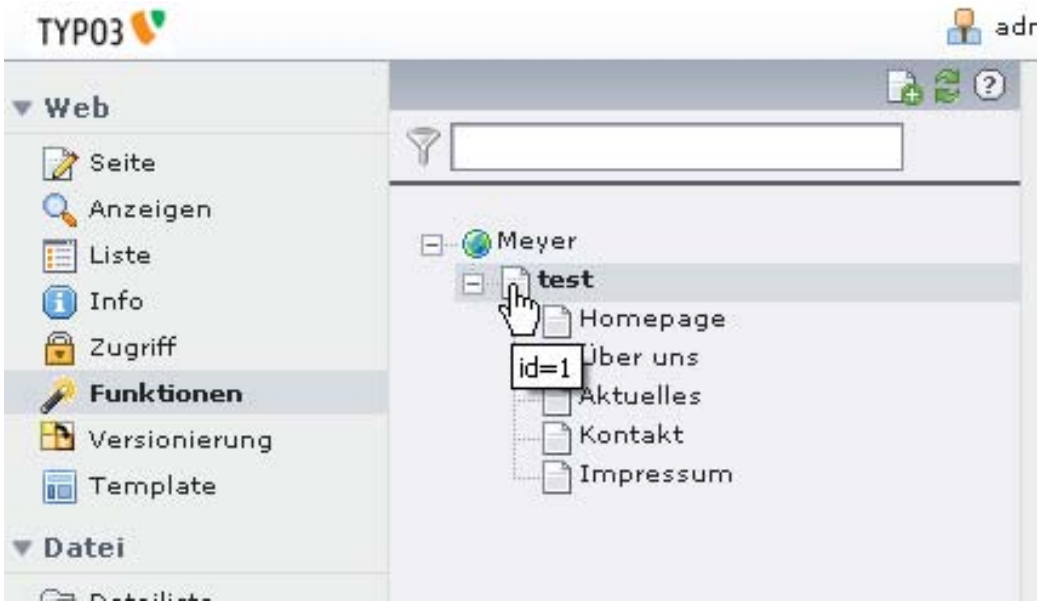
Dieses Beispiel ist aus zwei Gründen nicht ausführbar:

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. Fehlt diese Angabe, wird von der aktuellen Seite ausgegangen, was beim Navigieren schnell zu Verwirrungen führt.

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 4.13 sowie 4.14 werden aber noch wesentliche 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!

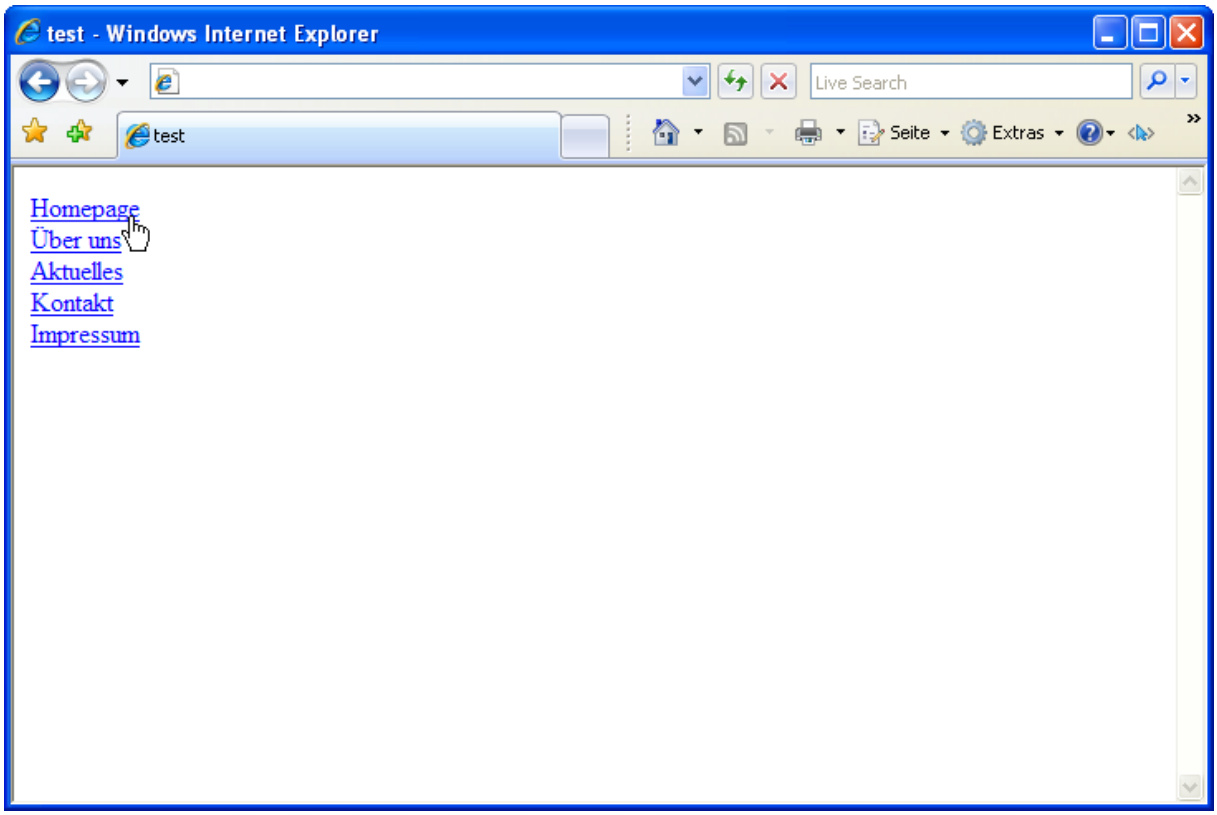
4.13 TMENU

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 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.

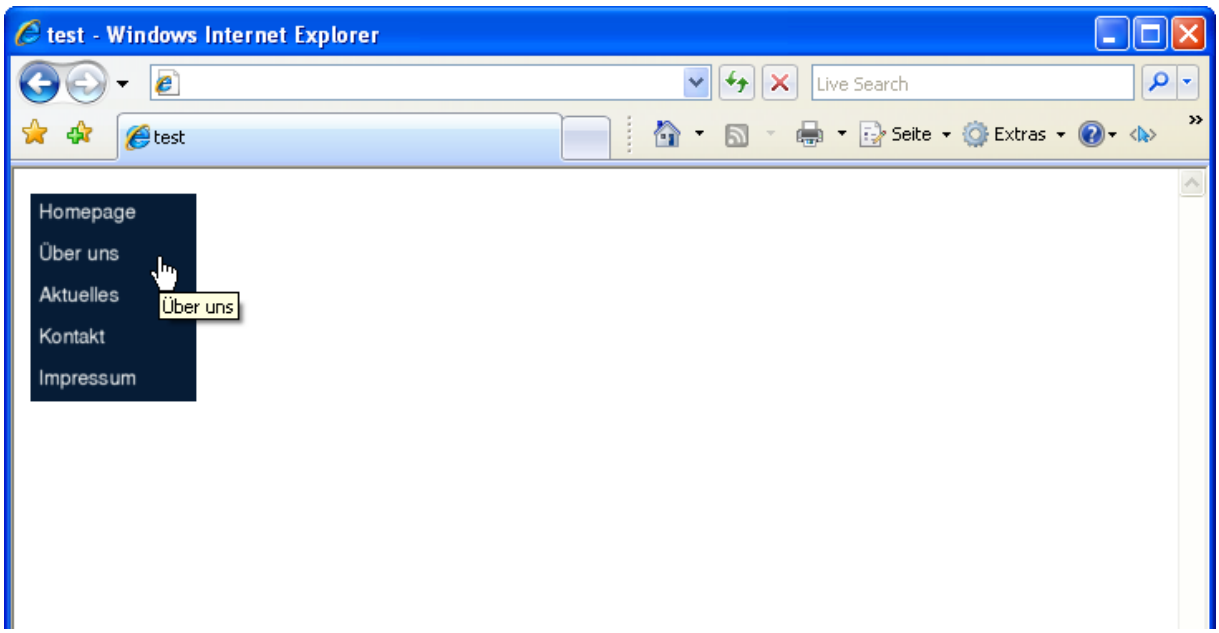
4.14 GMENU

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 = #001A32
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 Menü mit der Eigenschaft special = directory, Zeile 5, handelt und dass 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 4.11). 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“.

4.14.1 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.

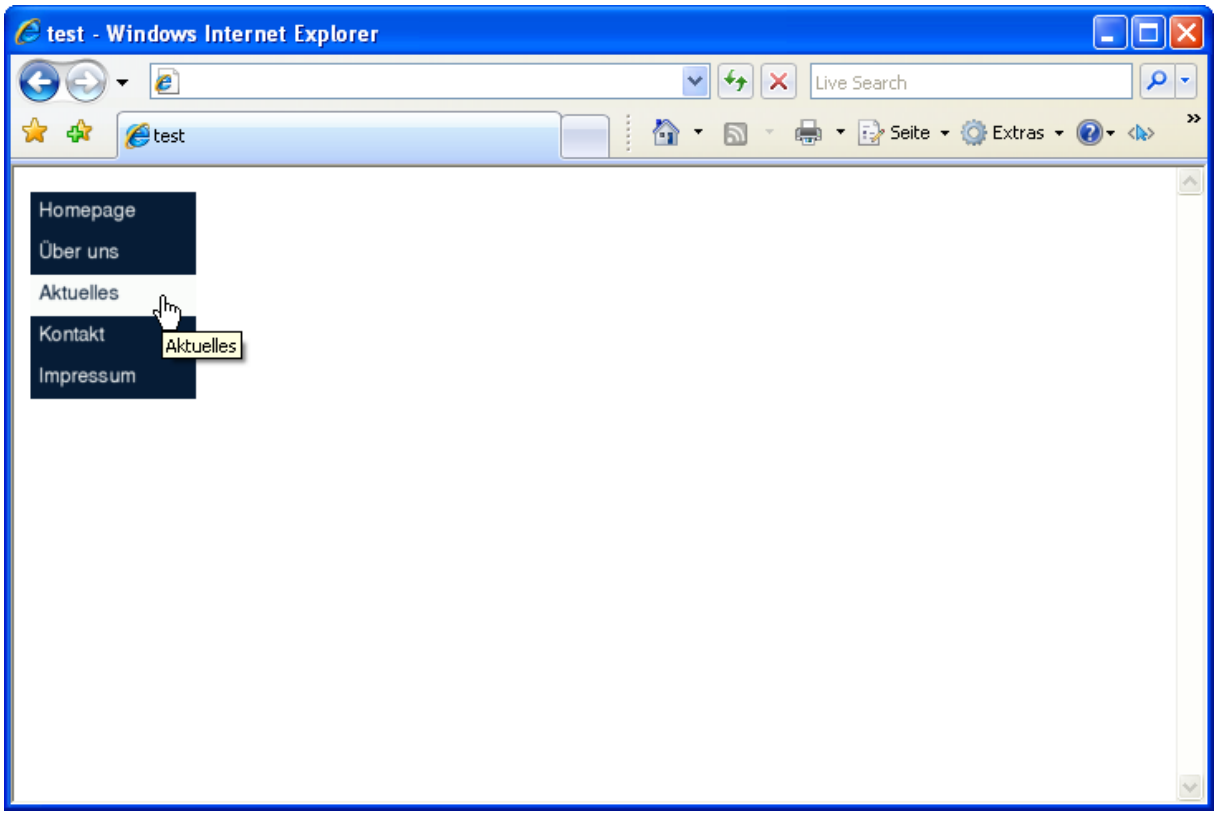
Beispiel 2

```
07 [...]
08      1 = GMENU
09      1.NO = 1
10      1.NO {
11            XY = 100, 25
12            backColor = #001A32
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 = white
24      1.RO.10.fontColor = #001A32
25 [...]
```

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. In Zeile 23 werden die Eigenschaft backColor geändert und in Zeile 24 die Schriftfarbe über fontColor.
- Wer in Zeile 9 den Zustand NO nicht aktivieren möchte (die Aktivierung des Zustandes NO ist optional), muss dann allerdings den Zustand RO explizit aktivieren, da die Wertzuweisung = 1 nicht mit kopiert werden würde:

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



4.14.2 Option Split: Elemente differenzieren

Mit so genannten 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.

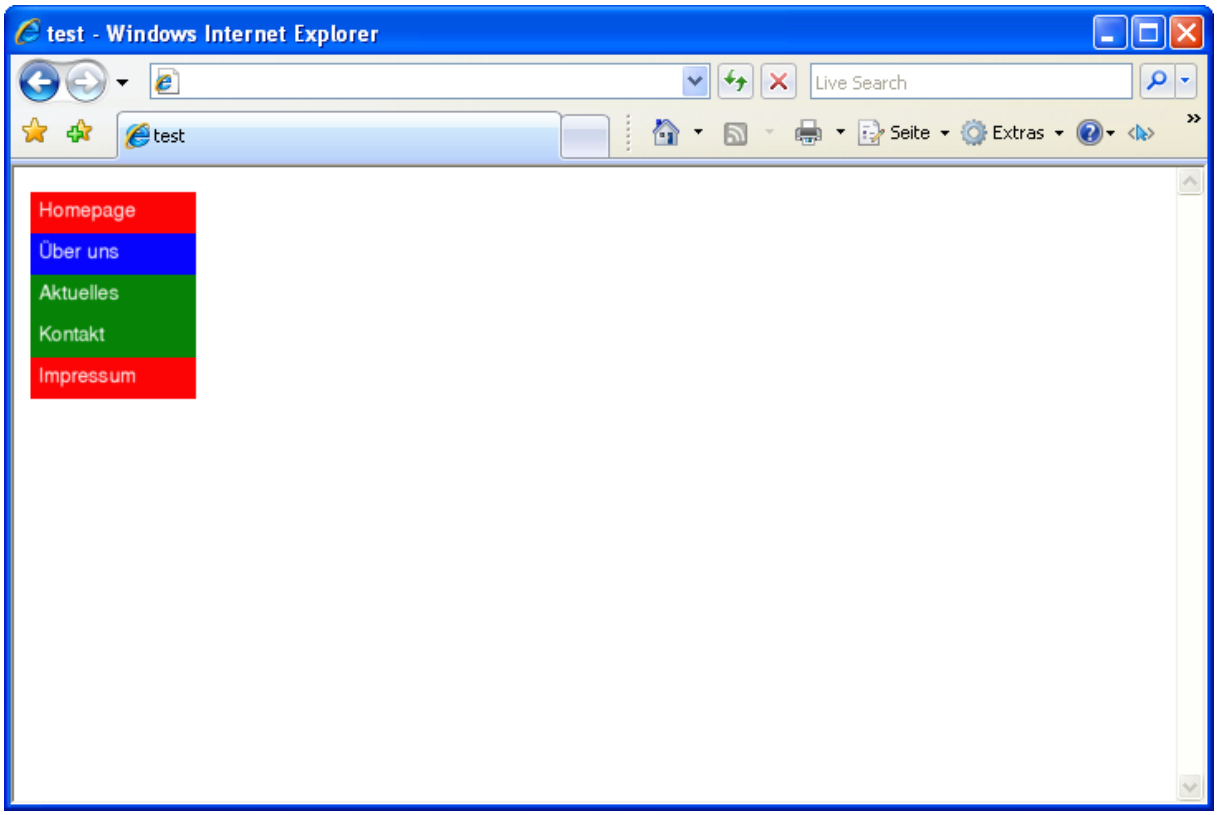
```
[...]
    1 = GMENU
    1.NO = 1
    1.NO {
        XY = 100, 25
        backColor = #001A32 |*| green |*| #001A32
        wrap = |<br>
        10 = TEXT
        10.text.field = title
        10.fontColor = white
        10.fontFile = fileadmin/fonts/verdanab.ttf
        10.fontSize = 12
        10.niceText = 1
        10.offset = 5, 15
    }
[...]
```

Das Symbol **|*|** unterteilt nicht nur in Anfang, Mitte und Ende, sondern auch in erstes, mittleres 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

```
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.


test - Windows Internet Explorer
Live Search
test
Seite
Extras
Homepage
Über uns
Aktuelles
Kontakt
Impressum

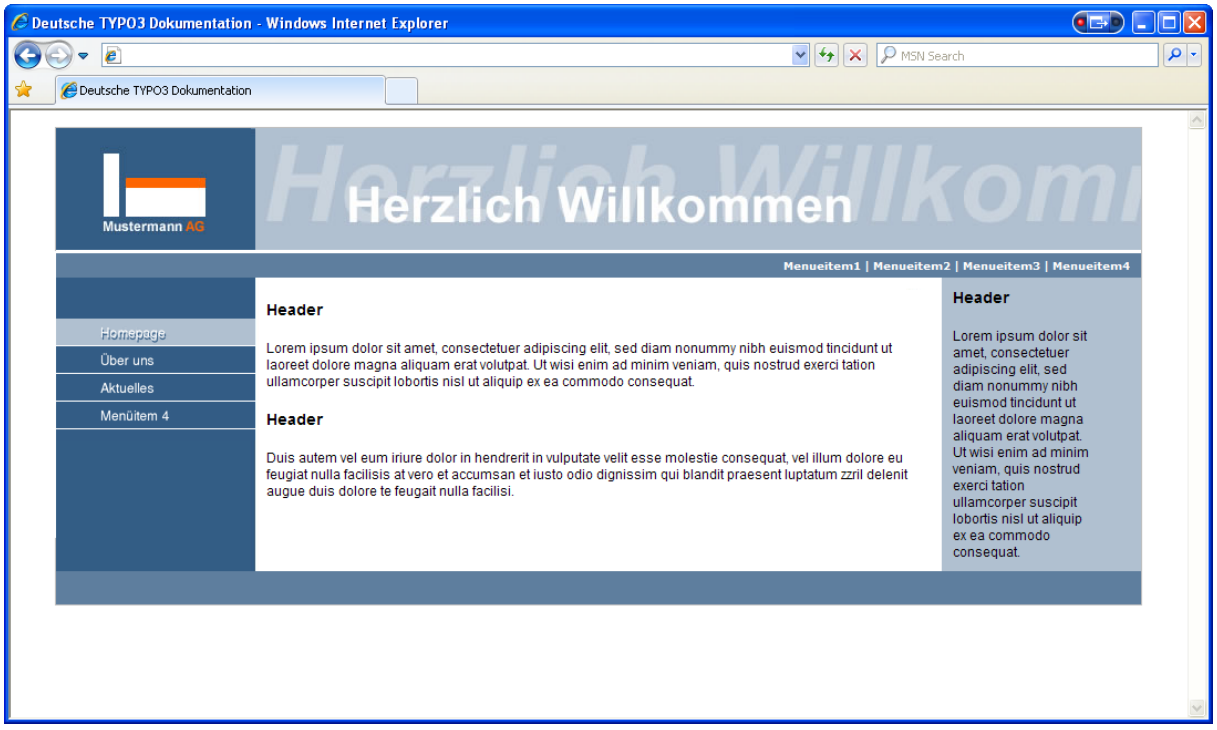
5. TypoScript Praxis

5.1 Vorwort

In diesem Kapitel 5 werden die Grundlagen aus Kapitel 4 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 links“ 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:



5.2 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 und welche Hilfsseiten werden benötigt?

2. Wie viele Ebenen wird die Navigation enthalten?

3. Sind Farbkonzepte vorgesehen?

4. Wo befindet sich die Homepage?

5.2.1 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 HMENUObjektes 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:

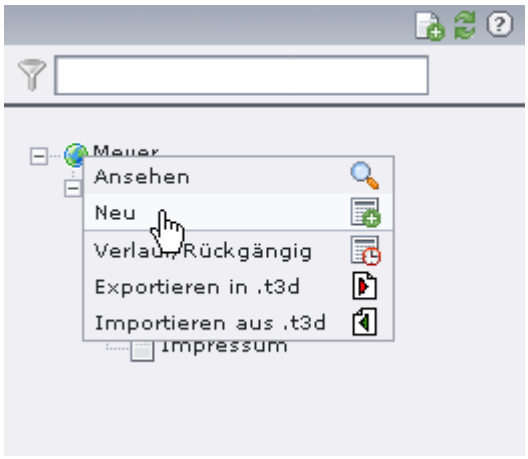
Ebene	Seite			Beschreibung
01	rootlevel			Root-Ebene (Weltkugel)
02		**Root**		Root Template (Vererbung) bzw. Startseite
03		**Menü Oben**		Hilfsmenüpunkt für Navigation
04			Homepage	Einzelnes Menüelement
04			Sitemap	Einzelnes Menüelement
04			Impressum	Einzelnes Menüelement
04			Kontakt	Einzelnes Menüelement
03		**Menü Links**		Hilfsmenüpunkt für Navigation
03			Homepage	Einzelnes Menüelement
03			Über Uns	Einzelnes Menüelement
03			Menüitem 1	Einzelnes Menüelement
03			Menüitem 2	Einzelnes Menüelement

03	Statisch Rechts	Hilfsmenüpunkt für permanent sichtbare Inhalte

5.2.2 Aufbau der Struktur im Frontend

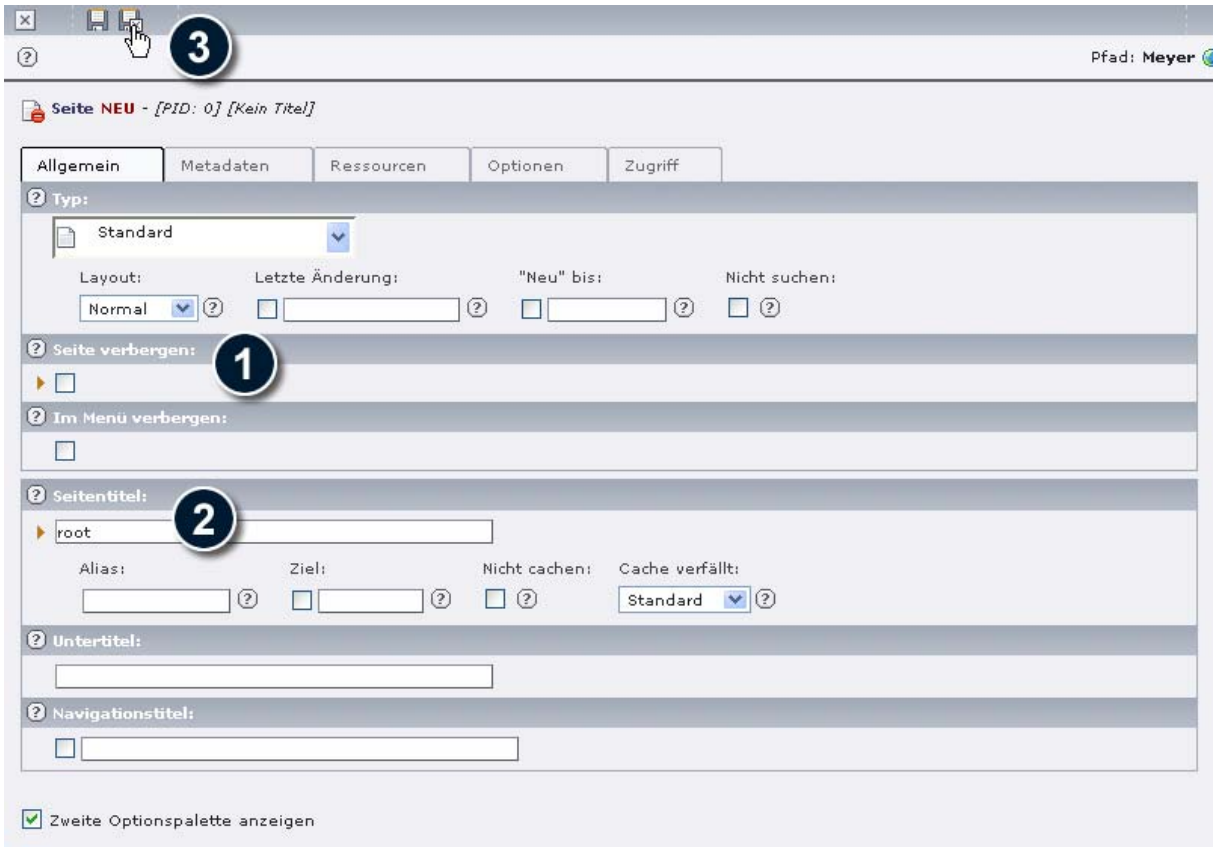
Obige Navigation soll nun in unserer TYPO3-Umgebung abgebildet werden. Die Seite, die wir in Kapitel 4 bereits angelegt haben, soll bestehen bleiben, aber keinen Einfluss auf unser Projekt haben.

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.



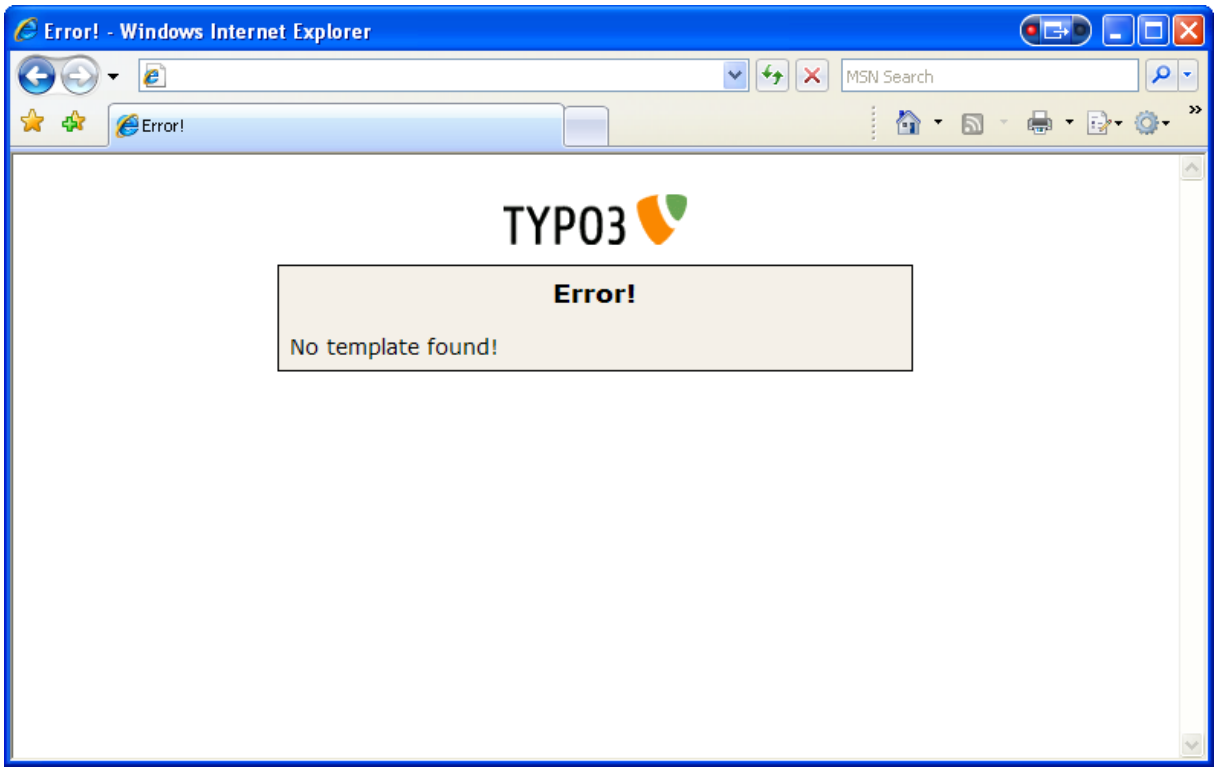
Auf der rechten Seite erscheint der Dialog „Neuer Datensatz“. Hier wählen wir den Eintrag „***Seite (in)***“ aus.

Es öffnet sich der Dialog für die Bearbeitung der Seiteneigenschaften, wo wir grundlegende Einstellungen für eine neue Seite angeben können. Da die Seite nicht versteckt sein soll, entfernen wir das Häkchen bei „***Seite verbergen***“ und geben als Seitentitel „***root***“ an. Über das Icon „***Speichern und Schließen***“ wird unsere neue Seite gespeichert und geschlossen.



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

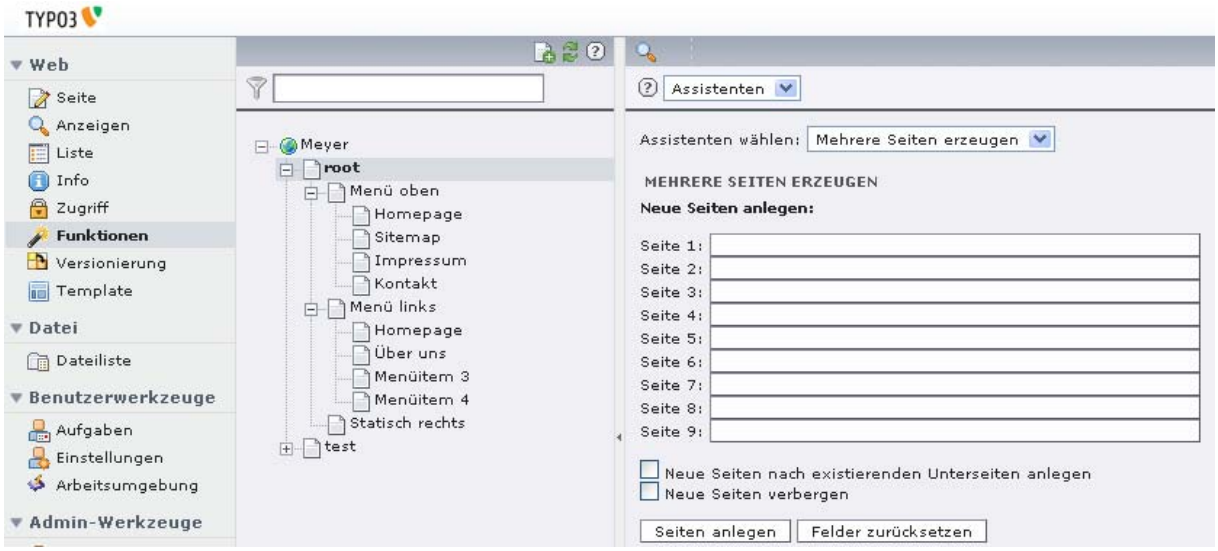
Betrachten wir nun unser (leeres) Projekt, indem wir auf die Weltkugel klicken und aus dem PopUp-Menü den Eintrag „***Anzeigen***“ auswählen, erhalten wir die Fehlermeldung „***No template found!***“.



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 dennoch die Seite „test“ betrachten zu können, kann 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.

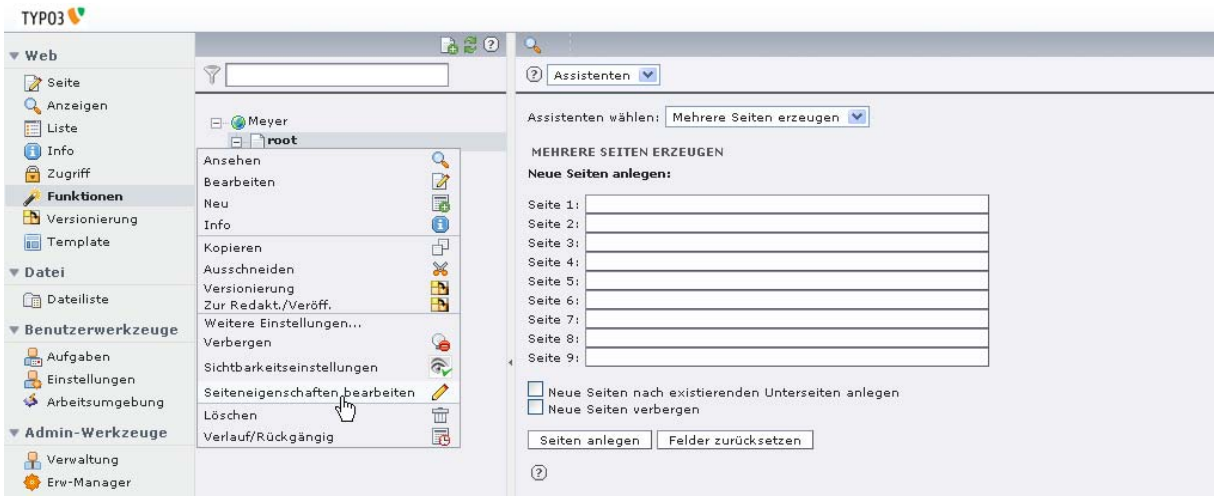


5.2.3 Hilfsseiten nicht zugänglich machen

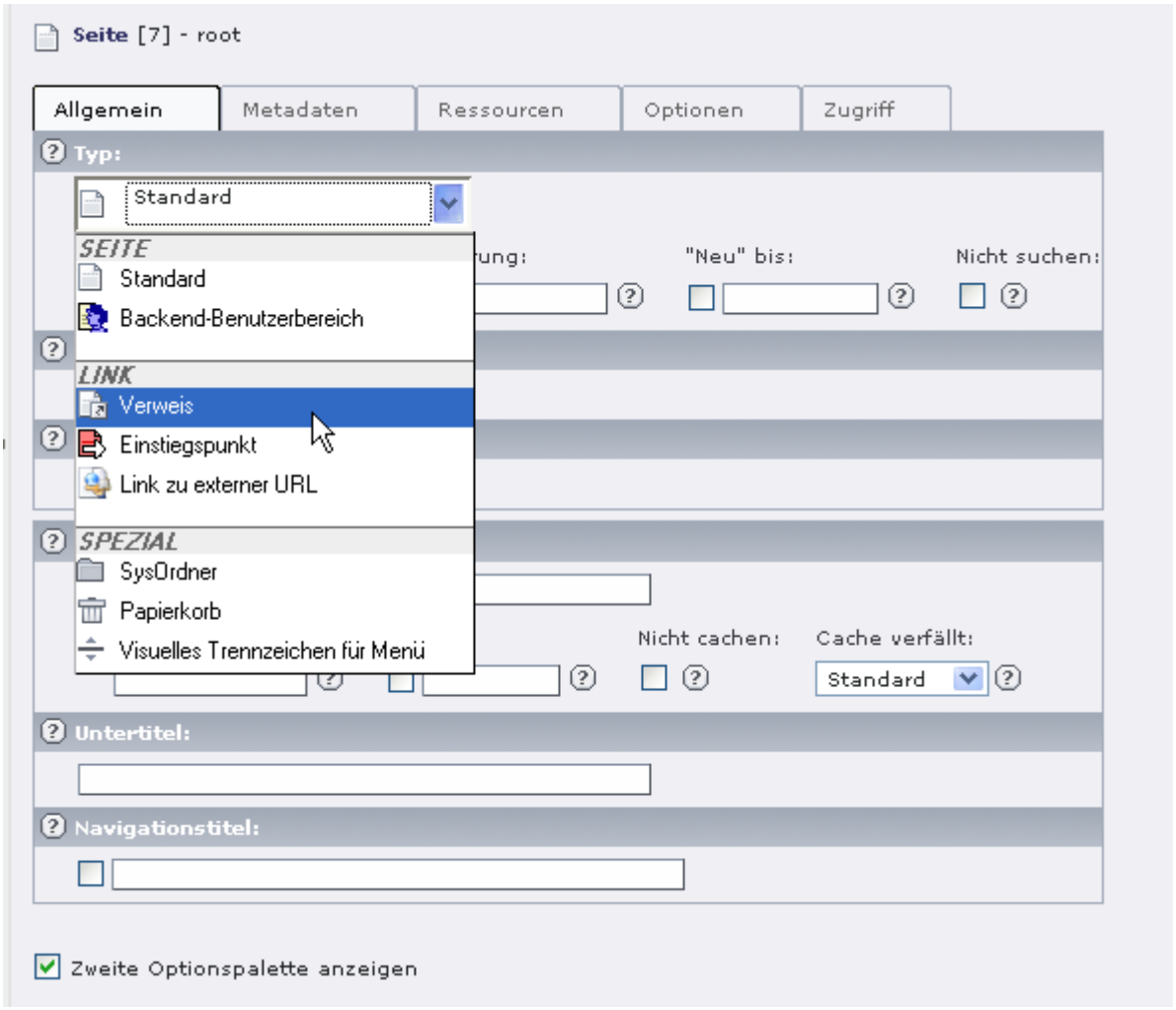
In unserer Struktur haben wir einige Seiten, die nicht zur Ablage von Inhalten bestimmt sind, dies sind im Detail die Seiten "root", "Menü oben", "Menü links" sowie "Statisch rechts". Diese Seiten gelten in unserem Projekt als "Hilfsseiten" und dienen zur Strukturierung sowie zur Template-Vererbung.

Beim Frontend-Aufruf ohne Angabe einer 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.

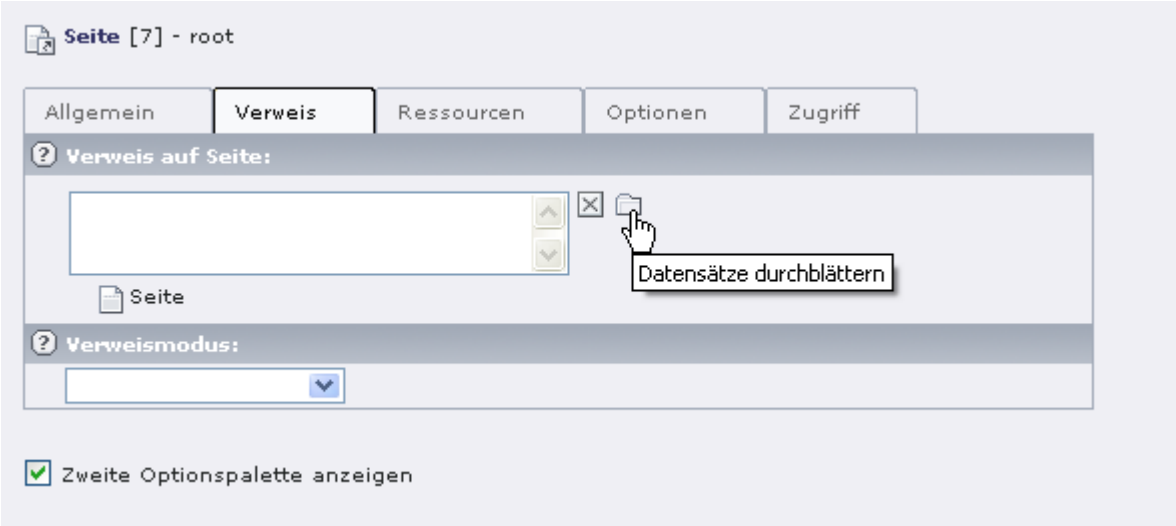
Diesen "***Verweis***" können wir erreichen, indem wir den Seitentyp von "***Standard***" auf "***Verweis***" setzen. Klicken Sie hierzu im Seitenbaum auf das Seitenicon vor der Seite "root" und wählen Sie aus dem PopUp-Menü den Eintrag "***Seiteneigenschaften bearbeiten***" aus.



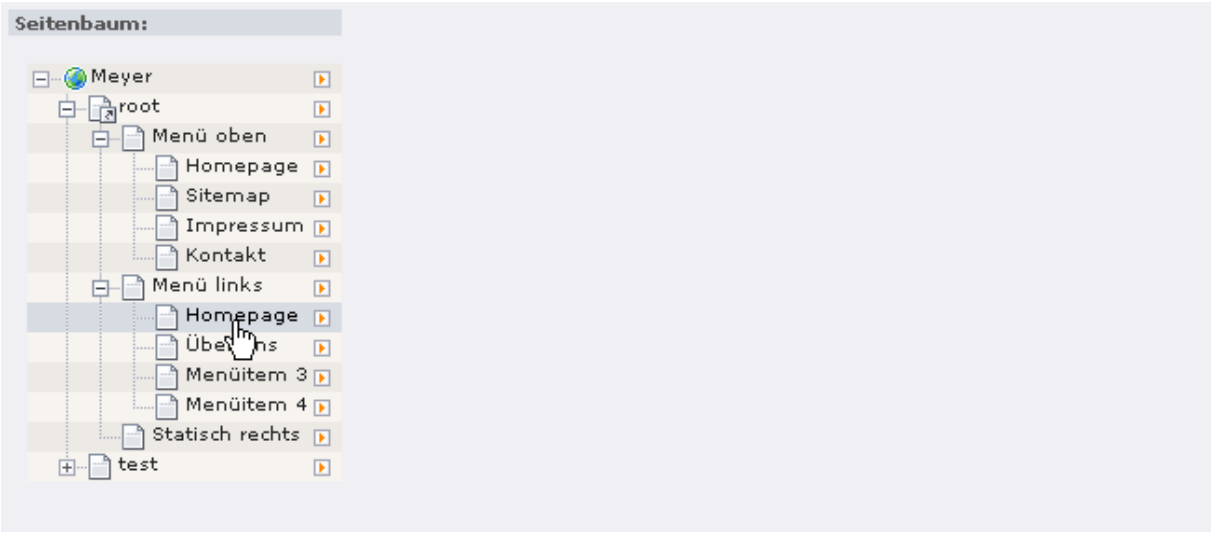
Auf der rechten Seite öffnet sich nun eine Maske in der die generellen Seiten-Eigenschaften bearbeitet werden können. Wählen Sie hier auf dem Reiter „***Allgemein***“ aus der DropDown-Box "***Typ***" den Eintrag „***Verweis***" aus.



Bestätigen Sie die folgende Meldung mit einem "***OK***". Die rechte Seite mit den Seiteneigenschaften wird neu geladen und es erscheint eine Maske mit einem neuen Reiter „Verweis“, in dem wir auswählen können, wohin der Verweis zeigt. Wechseln Sie auf den Reiter und klicken Sie auf das im Screenshot gezeigte Icon, um eine bestehende Seite auszuwählen, auf die der Verweis zeigen soll.



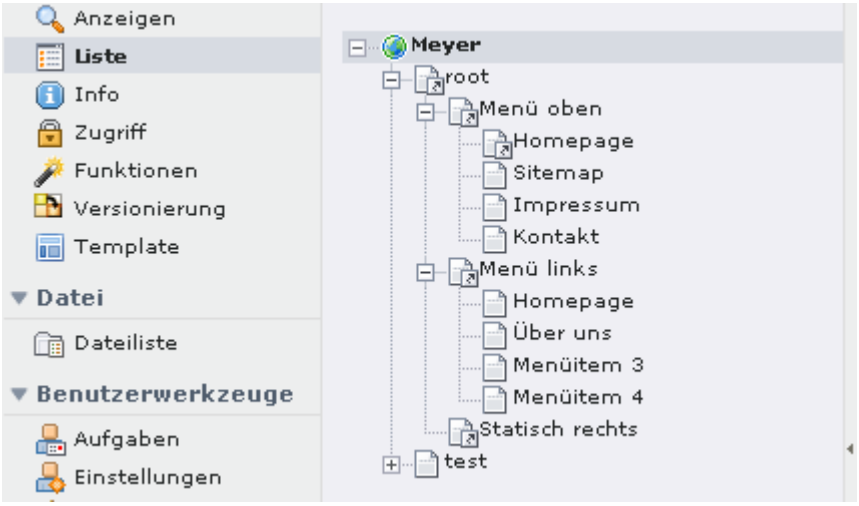
Wählen Sie aus dem sich öffnendem Datensatzbrowser die Seite "***Homepage***" aus, die sich im Abschnitt "***Menü links***" befindet.



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 Sie eine Fehlermeldung "**No propper Connection to the tree root**" im Frontend erhalten, überprüfen Sie bitte, ob Sie den Verweis richtig gesetzt haben. Wo befindet sich unsere Homepage?

Wiederholen Sie den obigen Schritt sowohl für die Seiten "Menü oben", "Menü links", "Statisch rechts" als auch für die Seite "Homepage", die sich im Abschnitt "Menü oben" befindet. Die Ansicht im Seitenbaum sieht anschließend wie folgt aus:



5.2.4 Wo befindet sich unsere Homepage?

Wir haben in unserer Seitenstruktur zweimal den Menüeintrag "Homepage" – einer wird jedoch praktisch benötigt. Im obigen Abschnitt haben wir bereits festgelegt, dass unsere eigentliche 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.

Der Vorteil 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.

5.3 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 oder TYPO3 Erweiterungen wie TemplaVoilà zu nutzen.

In diesem Kapitel arbeiten wir mehr mit unseren bewährten HTML-Tools als mit dem TYPO3-Backend.

5.3.1 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.

5.3.2 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\.

5.3.3 Substituieren von dynamischen Elementen

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

Statische Elemente unserer Präsentation:

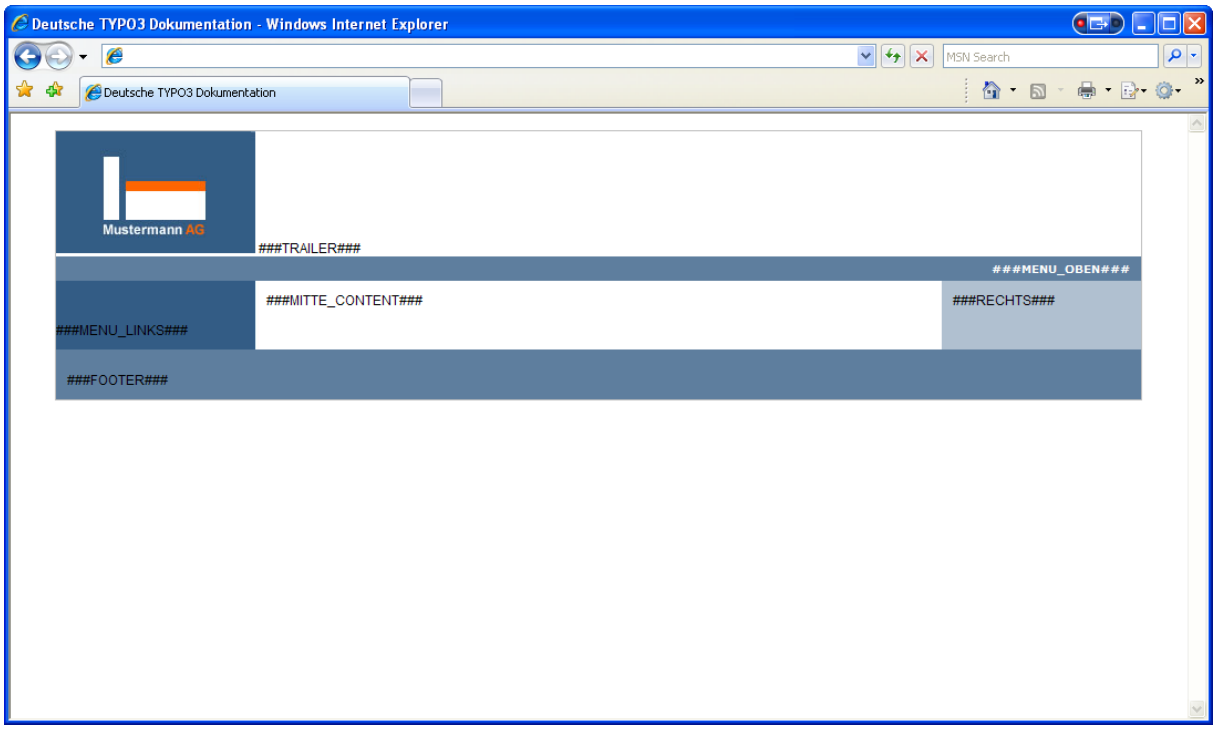
- Die generelle Containerstruktur und die durch die CSS Formatierungen
- Das Logo

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 - Marker (###BEZEICHNER###) ersetzt. Weiter werden alle Inhalte, die mehrfach auftreten, z.B. die Darstellung von Inhaltselementen im Content-Block, entfernt.

Wir ersetzen also in unserer Designvorlage alle Bereiche, die wiederkehrend sind, so dass ein „rudimentäres“ Design entsteht:



Der in unserer Designvorlage verwendete HTML-Code enthält keine Formatierungen:

```
01 <html>
02 <head>
03    <title>Deutsche TYPO3 Dokumentation</title>
04    <link rel="stylesheet" type="text/css" href="fileadmin/style.css" />
05 </head>
06 <body>
07    <!-- ###DOKUMENT### begin -->
08    <div id="container">
09          <div id="header">
10                <img src="fileadmin/images/logo.jpg">
11                ###TRAILER###
12          </div>
13          <div id="top">
14                ###MENU_OBEN###
15          </div>
16          <div id="links">
17                ###MENU_LINKS###
18          </div>
19          <div id="center">
20                ###MITTE_CONTENT###
21          </div>
22          <div id="right">
23                ###RECHTS###
24          </div>
25          <div id="footer">
26                ###FOOTER###
27          </div>
28    </div>
```

```
29    <!-- ###DOKUMENT### end -->
30    </body>
31 </html>
```

Und die zugehörige CSS Datei fileadmin/style.css:

```
01 h1{
02    font: normal 14px  Helvetica,Verdana,Arial;
03    font-weight: bold;
04 }
05
06 div,img{
07    margin:0;
08    padding:0;
09    border:0;
10 }
10
11 body{
12    text-align:center;
13 }
14
15 div#container{
16    width:980px;
17    text-align:left;
18    margin:0px auto;
19    background:  url(../fileadmin/images/background.jpg)     repeat-y;
20 }
21
22 div#top   {
23    font: bold 10px    Verdana,Arial;
24    clear:both;
25    background-color:#5E7E9E;
26    text-align:right;
27    color:#FFFFFF;
28    padding:5px;
29    padding-right:10px;
30 }
31
32 a.linkWeiss{text-decoration:none;color:white;}
33
34 a:hover.linkWeiss{text-decoration:underline;}
35
36 div#header{background-color:white;}
37
38 div#links{
39    float:left;
40    width:180px;
41    padding-top:37px;
42    padding-bottom:10px;
43    background-color:#335C85;
44 }
45
```

```
46 div#center {float:left;width:600px;padding:10px;}
47
48 div#head {
49    font: normal 10px Helvetica,Verdana,Arial;
50    text-align:right;
51 }
52
53 div#right{float:left;width:160px;height:auto;padding:10px;}
54
55 div#footer {
56    clear:both;
57    background-color:#5E7E9E;
58    padding:4px;
59    padding-bottom:2px;
60 }
61
62 div#footer a{text-decoration:none;color:white;}
63
64 div#footer a:hover{text-decoration:underline;}
65
66 div#links_bottom{
67    text-align:center;
68    background-color:#8BA2BA;
69    margin-top:20px;
70    padding-top:1px;
71    padding-bottom:1px;
72 }
73
74 .float-left{float:left;}
75
76 /* Anpassungen IE     */
77
78 *html div#container{width:982px;}
79
80 *html div#center{float:left;width:620px;}
81
82 *html div#footer,div#top{width:980px;}
```

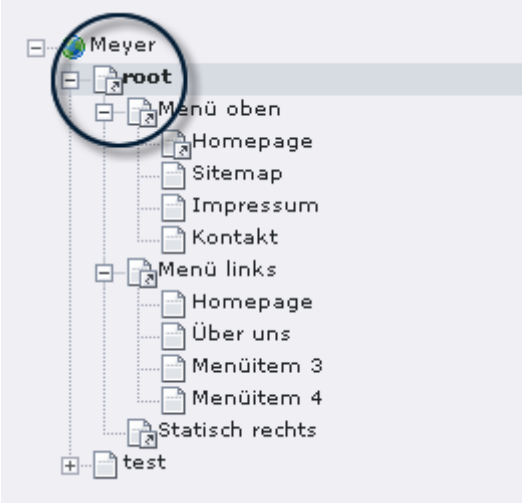
Von unserem anfänglich erstellen HTML-Dokument ist nicht mehr besonders viel übriggeblieben. Das anfangs erstellte Dokument sollte jedoch auf allen gewünschten Browsern in der gewünschten Qualität zur Verfügung stehen. Spätere Korrekturen sind zwar immer noch möglich – in der Praxis wird die Ursache jedoch häufig in TYPO3 gesucht und nicht an der Designvorlage selbst.

5.4 Umsetzung mit TypoScript

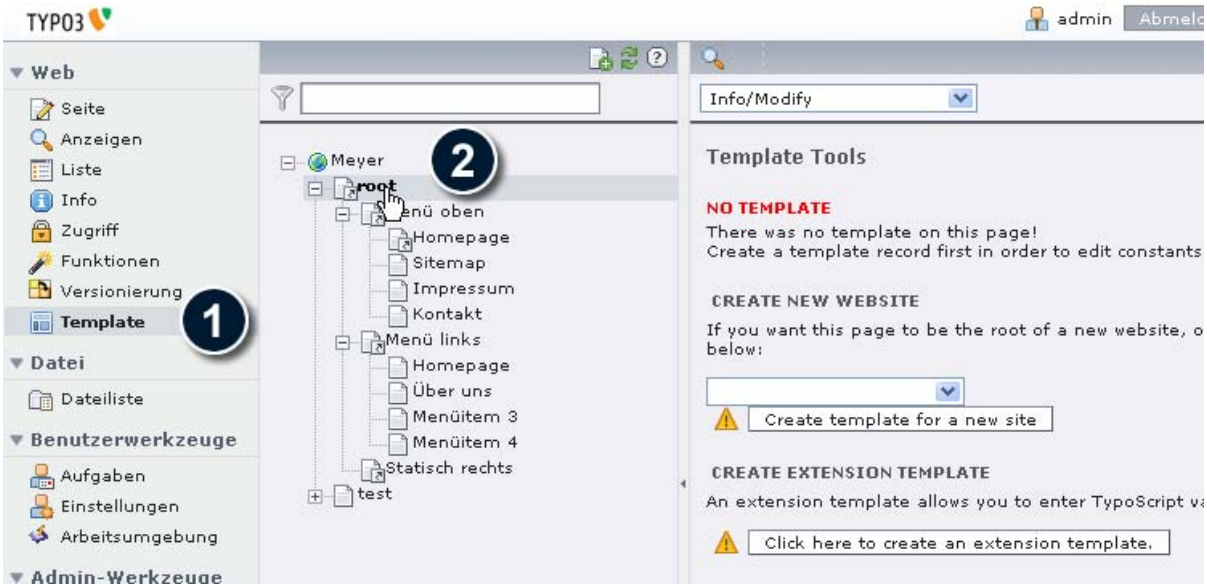
5.4.1 Das Root-Template erstellen

Im ersten Schritt legen wir ein Root-Template unserer Seite „root“ an. Dieses hier angelegte Template wird vererbt und ist somit auch für die Unterseiten gültig.

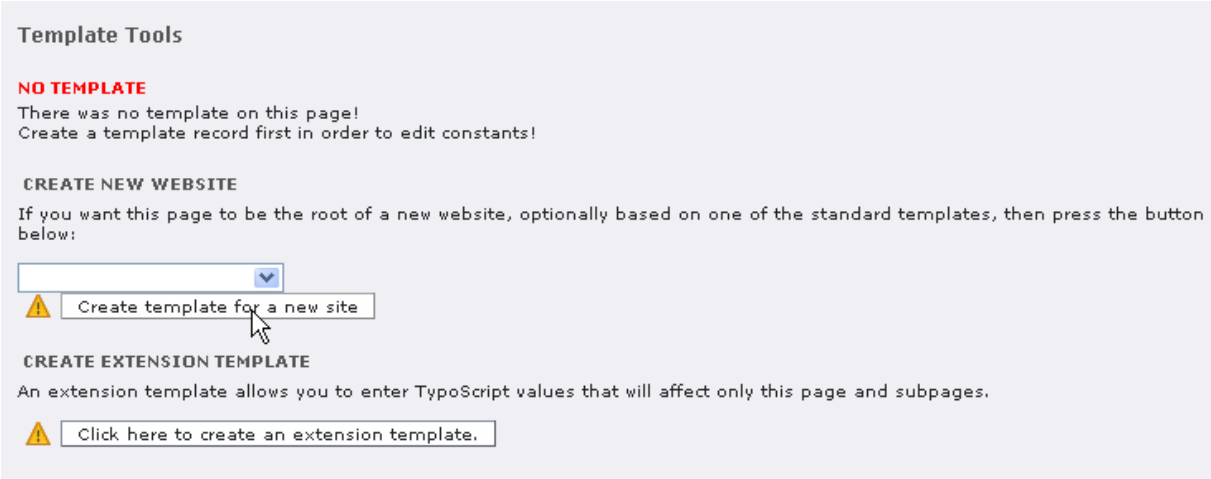
Wie gewünscht ist das in Kapitel 4 angelegte Template auf der Seite „test“ von unserem neuen Template nicht betroffen, da sich die Seite „test“ auf der gleichen Ebene befindet wie unsere Seite „root“.



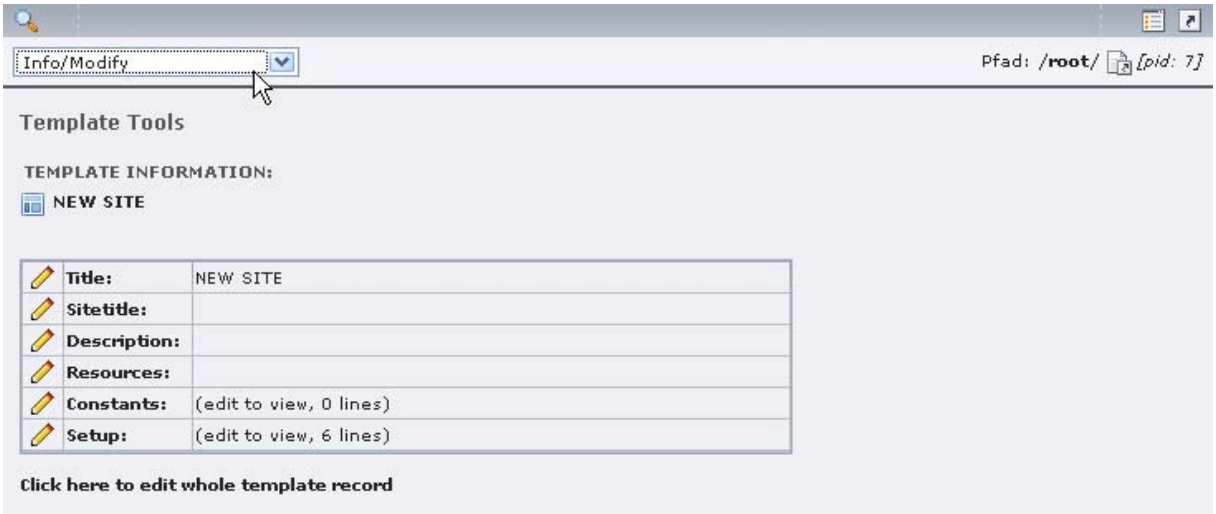
Zum Anlegen eines Templates wählen wir auf der Menüliste auf der rechten Seite den Eintrag „Template“ aus „Aktivieren“ dann unsere Seite „root“ (auf die das neue Template angelegt werden soll), indem wir auf den Textlink klicken.



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.



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, das Feld „Setup“ zu editieren. Hierzu klicken wir auf das Bleistiftsystem vor dem Feld „***Setup***“.

Im Feld **„Setup“** können wir unser zu erstellenden **TypoScript-Code** ablegen

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

5.4.2 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.

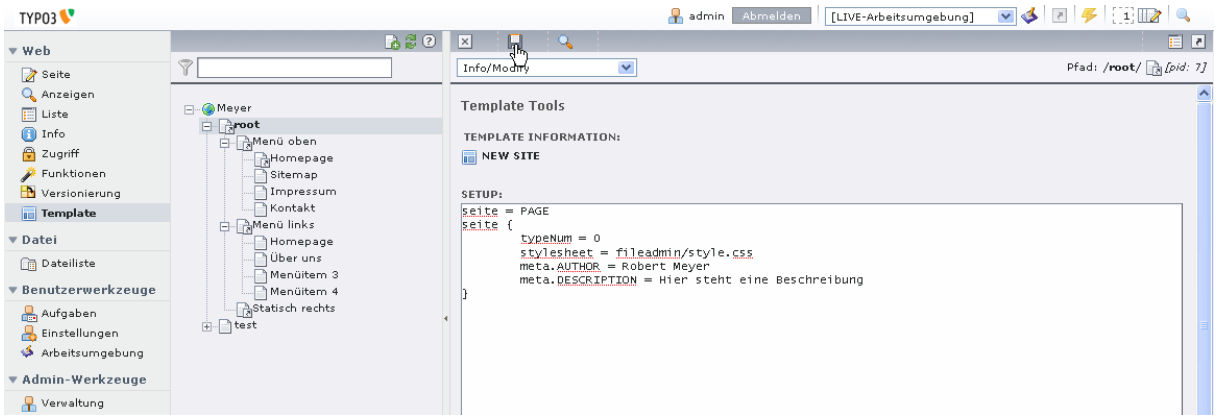
Beispiel 1:

```
01 seite = PAGE
02 seite {
03     typeNum = 0
04     stylesheet = fileadmin/style.css
05     meta.AUTHOR = Robert Meyer
06     meta.DESCRIPTION = Hier steht eine Beschreibung
07 }
```

- 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 7. "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 musste in älteren TYPO3-Versionen zwingend gesetzt werden. Der Vollständigkeit halber und zum besseren Verständnis setzen wir diese Eigenschaft hier auch. In einem Template muss es genau eine PAGE-Instanz mit typeNum = 0 geben. Nähere Informationen zur typeNum-Eigenschaft finden Sie im Kapitel 4.1
- In Zeile 4 wird unser Stylesheet eingelesen. Sofern die Datei nicht unter dem angegebenen Pfad gefunden wird, wird das <style> Tag nicht im Frontend ausgegeben. Achten Sie daher auf den richtigen Pfad.
- In den Zeilen 5 und 6 werden MetaTags statisch angegeben. Aus Sicht der Suchmaschinen-Optimierung sollten diese jedoch dynamisch gehalten werden.

In dem Beispiel in dieser Dokumentation haben wir als Bezeichner ***seite*** für das TLO Objekt PAGE verwendet. Sie können hier einen beliebigen Bezeichner wählen, daher habe ich absichtlich nicht das Wort ***page*** als Bezeichner verwendet. Einige Extensions gehen jedoch von dem Bezeichner ***page*** als Standard aus, daher sollte Sie page = PAGE im Life-Betrieb nutzen.

Bearbeiten Sie das Setupfeld des Root Templates über das Bleistift-Icon. Fügen Sie den TypoScript Code vom Beispiel 1 in das Setup Feld ein und speichern Sie die getätigten Eingaben.



Über das Lupen-Icon können Sie das Frontend aufrufen. Ein Betrachten dieses Templates im Frontend liefert eine leere Seite zurück. Ein Betrachten des HTML-Quelltextes zeigt, dass unsere Zuweisungen bereits gefruchtet haben:

```
01 <!DOCTYPE html
02     PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
03 <html>
04 <head>
05     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
06 <!--
07     This website is powered by TYPO3 - inspiring people to share!
08     TYPO3 is a free open source Content Management Framework initially
09     created by Kasper Skaarhoj and licensed under GNU/GPL.
10     TYPO3 is copyright 1998-2008 of Kasper Skaarhoj. Extensions are
11     copyright of their respective owners.
12     Information and contribution at http://typo3.com/ and
13     http://typo3.org/
14     -->
15
16     <title>Homepage</title>
17     <meta name="generator" content="TYPO3 4.2 CMS" />
18     <meta name="AUTHOR" content="Robert Meyer" />
19     <meta name="DESCRIPTION" content="Hier steht eine Beschreibung" />
20     <script type="text/javascript"
       src="typo3temp/javascript_93077bb238.js"></script>
21
22 </head>
23 <body>
24
25 </body>
26 </html>
```

In Zeile 16 wird der Seitentitel unserer betrachteten Seite "homepage" ausgegeben. Da die Seite „root“ ein Verweis auf die Seite „Homepage“ ist, wird der Titel der Seite „Homepage“ ausgegeben Dieses wird von TYPO3 selbstständig getätigt.

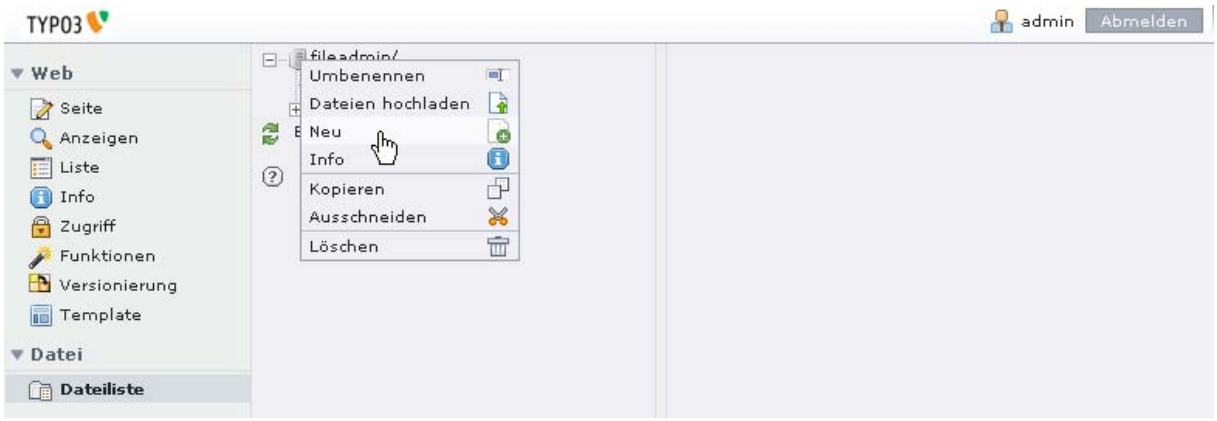
Durch Angabe eines "Sitetitles" im Root Template kann der Titel um eine statische Komponente erweitert werden (z.B. "Mustermann AG: homepage"). Nähere Informationen hierzu finden Sie im Kapitel 3.

5.4.3 Dateien mittels Dateimanager zur Verfügung stellen

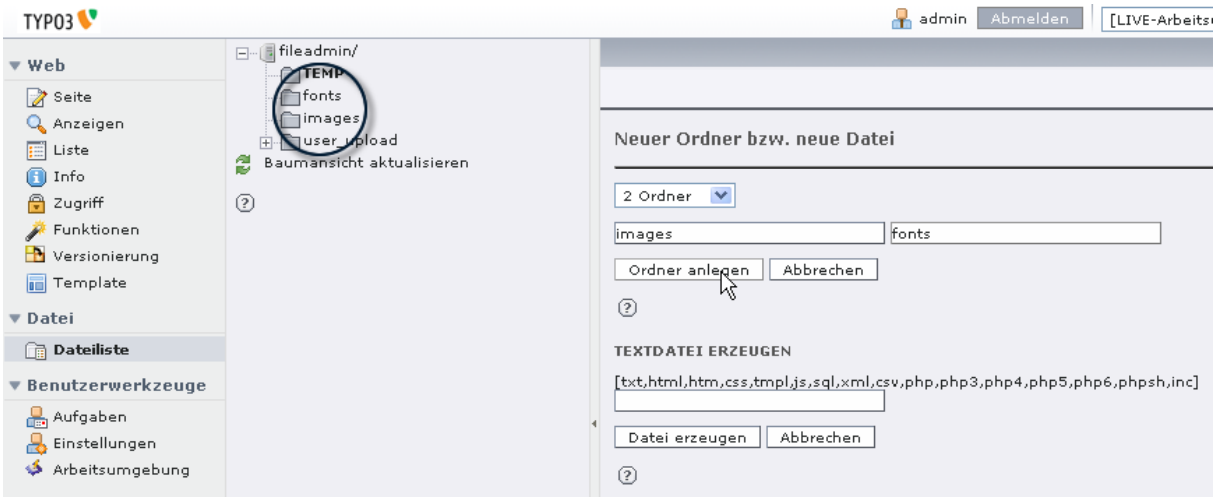
Im Kapitel 5.3 haben wir unsere Designvorlage erstellt. Damit diese Designvorlage in unserer TYPO3-Umgebung zur Verfügung steht, muss sie zunächst auf unseren Server hochgeladen werden. Dieses ist prinzipiell mit FTP möglich, hier soll jedoch das Hochladen mittels des integrierten Dateimanagers gezeigt 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 benötigte 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.

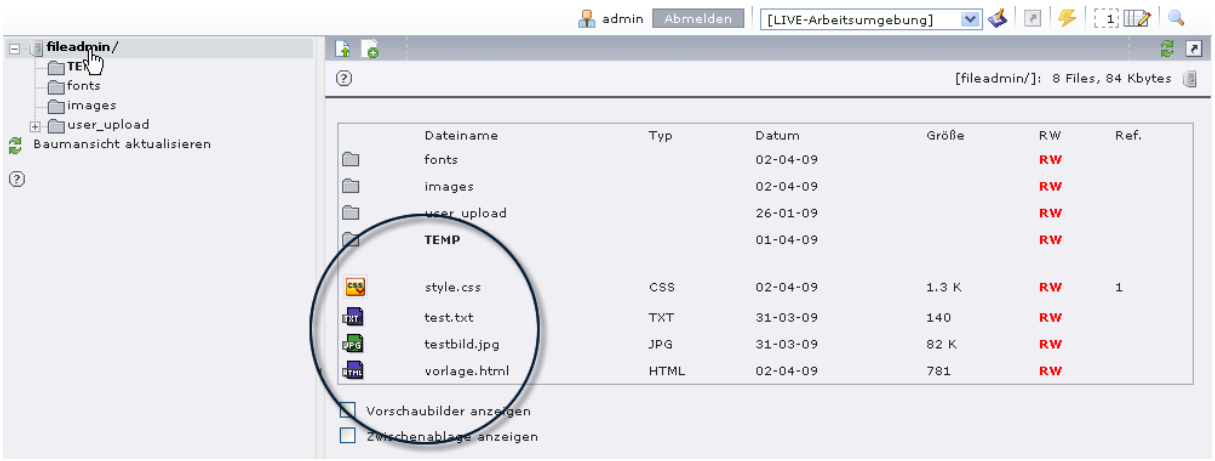


Erstellen Sie nun unterhalb von "fileadmin" zwei Ordner: "***images***" und "***fonts***".



Um nun unsere Designvorlage "vorlage.html" im System zur Verfügung zu stellen (Datei hoch laden), klicken Sie im Verzeichnisbaum auf das Icon vor "fileadmin" und wählen Sie aus dem sich öffnendem PopUp-Fenster den Menüeintrag "Datei hochladen " 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 Dateien "logo.jpg". und „background.jpg“ 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, arialbd.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 kleingeschrieben wird (TypoScript ist Case-Sensitive).

5.4.4 Die Designvorlage einbinden

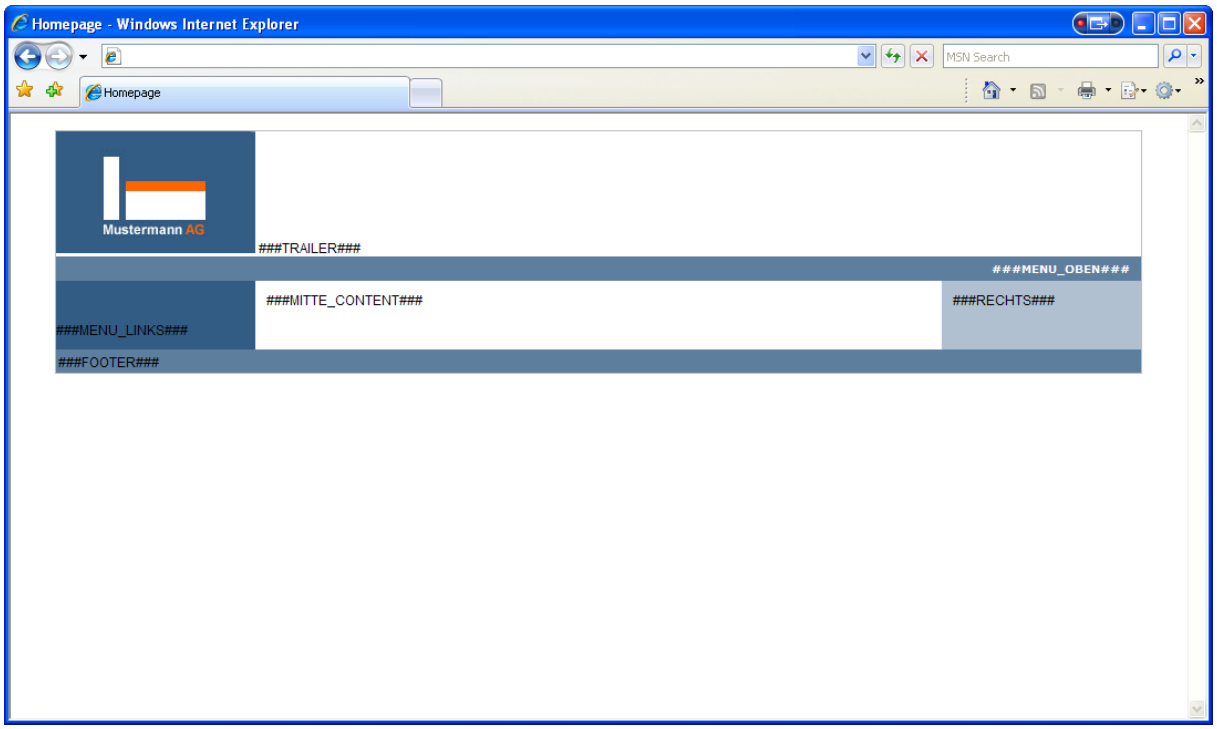
Nachdem wir die Designvorlage nun im Ordner "fileadmin" zur Verfügung gestellt haben, können wir diese auch über TypoScript ansprechen. Hierzu erweitern wir unser Beispiel 1:

Beispiel 2:

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

- In Zeile 8 wird "an der Position 10" eine Instanz des TEMPLATE-Objektes erzeugt.
- In Zeile 9 wird angegeben, dass die Vorlage aus einer Datei stammen soll. Der genaue Ort dieser Designvorlage wird in Zeile 10 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:

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
02 <html>
03 <head>
04    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
05    <!--
06          This website is powered by TYPO3 - inspiring people to share!
07          [...]
08    -->
09 </head>
10 <body>
11 <html>
12    <head>
13     <title>Deutsche TYPO3 Dokumentation</title>
14     <link rel="stylesheet" type="text/css" href="fileadmin/style.css" />
15    </head>
16    <body>
17          <!-- ###DOKUMENT### begin -->
18
19          <div id="container">
```

In den Zeilen 09 bis 16 können wir den Fehler erkennen: HTML-, Header- und BodyTags sind doppelt definiert. Bis einschließlich Zeile 11 wurde der HTML-Quelltext direkt von TYPO3 aus erzeugt. Ab der Zeile 12 wurde unsere Designvorlage eingelesen – incl. aller dort gespeicherten HTML-Tags, wie z.B. der <head>-Abschnitt und auch der BodyTag.

In Zeile 17 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:

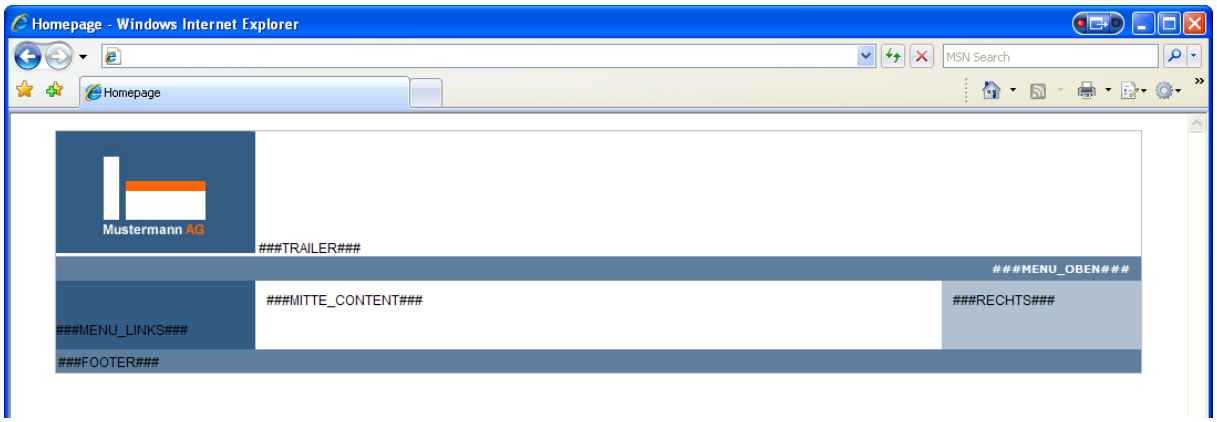
```
01 seite = PAGE
02 seite {
       [...]
08     10 = TEMPLATE
09     10.template = FILE
10     10.template.file = fileadmin/vorlage.html
11     10.workOnSubpart = DOKUMENT
12 }
```

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

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
02 <html>
03 <head>
04    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
05    <!--
06          This website is powered by TYPO3 - inspiring people to share!
07          [...]
08    -->
09 </head>
10 <body>
11 <html>
12    <div id="container">
13
14          <div id="header">
15                <img src="fileadmin/images/logo.jpg">
16                ###TRAILER###
17          </div>
18
19          <div id="top">
20                [...]
```

5.4.5 Die Platzhalter ansprechen: Fehleranalyse

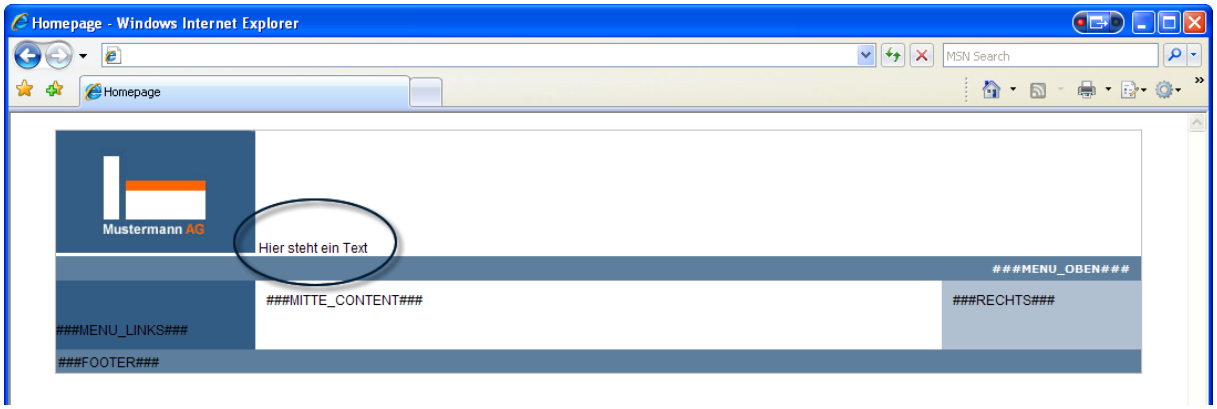
In diesem Abschnitt werden Tipps und Tricks zur Fehleranalyse gegeben. Beim Betrachten der Präsentation im Frontend werden alle verfügbaren Marker angezeigt.



Diese sollen Schritt für Schritt ersetzt werden. Das Ersetzen eines Platzhalters/Markers geschieht mittels der Eigenschaft "***marks***":

```
01 seite = PAGE
02 seite {
       [...]
08     10 = TEMPLATE
09     10.template = FILE
10     10.template.file = fileadmin/vorlage.html
11     10.workOnSubpart = DOKUMENT
12     10.marks.TRAILER = TEXT
13     10.marks.TRAILER.value = Hier steht ein Text
14}
```

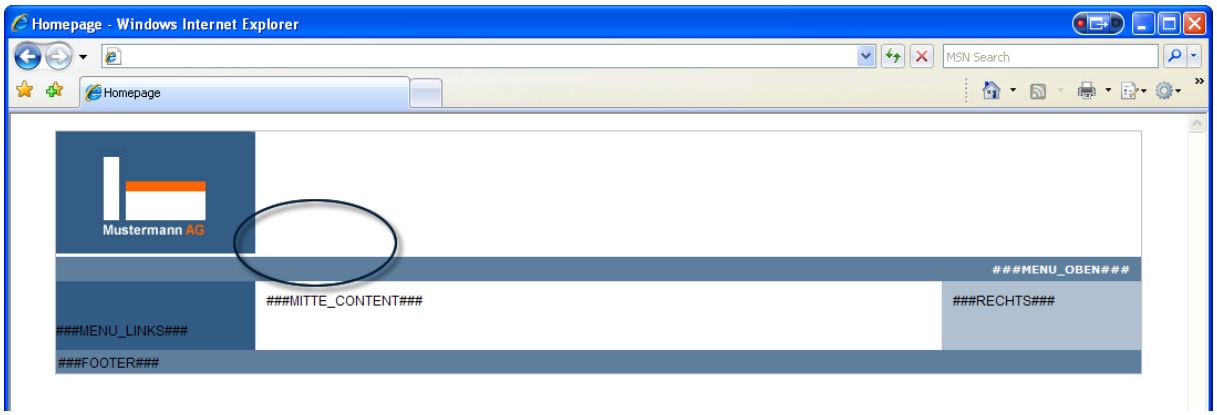
Im Frontend können wir sehen, dass der Marker angesprochen und auch schon das gewünschte Ergebnis zurückgeliefert wurde. Natürlich soll im Trailer im nächsten Abschnitt dynamisch eine Grafik eingelesen werden.



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

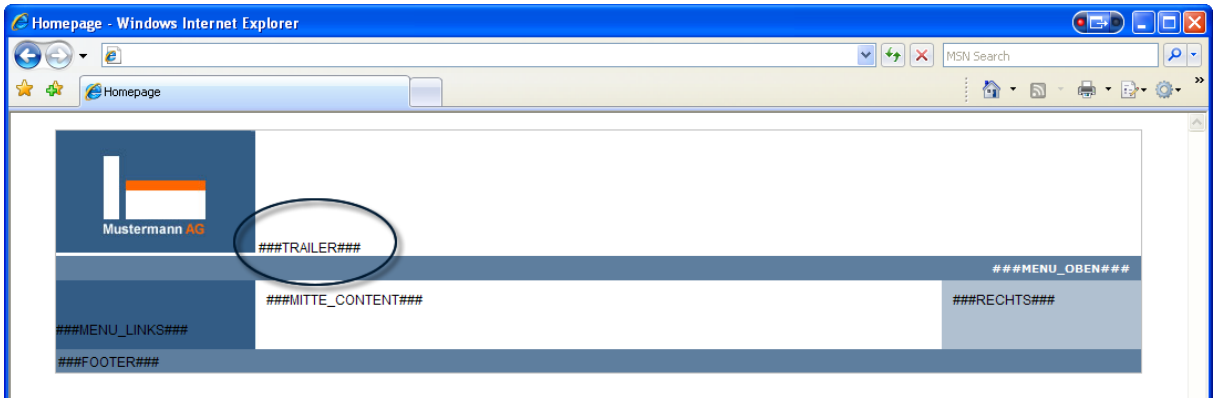
```
01 seite = PAGE
02 seite {
       [...]
08     10 = TEMPLATE
```

```
09     10.template = FILE
10     10.template.file = fileadmin/vorlage.html
11     10.workOnSubpart = DOKUMENT
12     10.marks.TRAILER = TEXT
13     10.marks.TRAILER.valuee = Hier steht ein Text
14 }
```



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

```
01 seite = PAGE
02 seite {
       [...]
08     10 = TEMPLATE
09     10.template = FILE
10     10.template.file = fileadmin/vorlage.html
11     10.workOnSubpart = DOKUMENT
12     10.marks.TREILER = TEXT
13     10.marks.TRAILER.value = Hier steht ein Text
14 }
```



Zusammenfassung:

- Der Marker wurde richtig angesprochen, wenn der Bezeichner im Frontend nicht mehr sichtbar ist (z.B. ###TRAILER###). Dies bedeutet also, dass die TypoScript Zeile 12 bis zum Gleichheitszeichen korrekt sein muss ("10.marks.TRAILER =")
- Ist der Bezeichner im Frontend noch sichtbar, wurde der Marker nicht korrekt angesprochen. Der Fehler ist **vor** dem Gleichheitszeichen in Zeile 12 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.

5.4.6 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 hellblaue Hintergrundfläche mit dem Farbcode #B0C0D0 und den definierten Abmaßen 797 x 110 Pixel
- Eine große, kursive Überschrift (ca. 100 Punkt), die auf der Hintergrundebene liegt. Schriftfarbe: #C8D3DE
- Eine etwas kleinere Überschrift (ca. 45 Punkt), die mitten auf der obersten Ebene liegt. Schriftfarbe: #ffffff

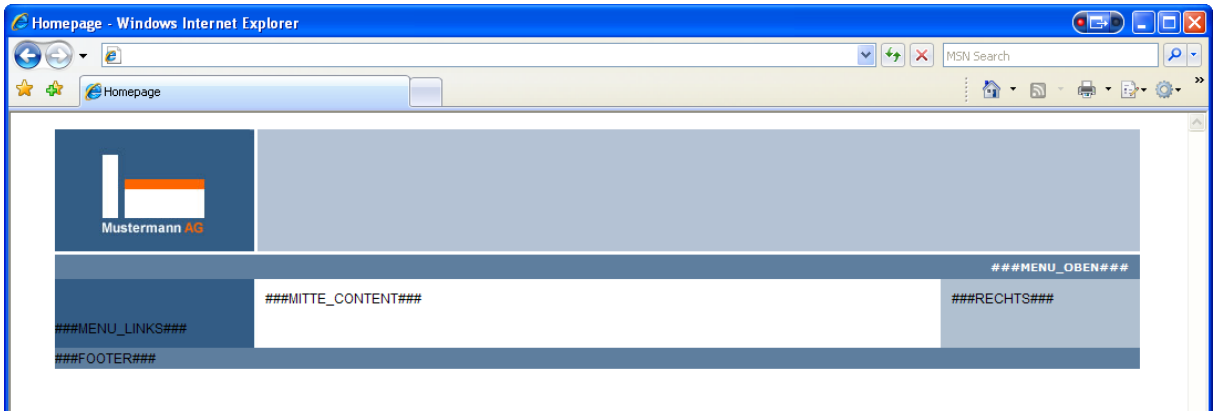
Dieses kann direkt in TypoScript abgebildet werden.

Im ersten Schritt kümmern wir uns zunächst darum, dass eine hellblaue Grafik mit entsprechenden Abmaßen dargestellt wird. Also müssen wir wieder das Setup unseres Root-Templates bearbeiten und folgende Zeilen ergänzen:

Beispiel 3

```
01 seite = PAGE
02 seite {
```

```
      [...]
08    10 = TEMPLATE
09    10.template = FILE
10    10.template.file = fileadmin/vorlage.html
11    10.workOnSubpart = DOKUMENT
12    10.marks{
13          TRAILER = IMAGE
14          TRAILER.file = GIFBUILDER
15          TRAILER.file{
16                XY = 797, 110
17                backColor = #B0C0D0
18          }
19    }
14}
```



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

- In Zeile 12 wird 10.marks ausgeklammert. Dies sorgt für ein übersichtlicheres Template
- In Zeile 13 wird auf dem Marker "TRAILER" eine IMAGE-Instanz gebildet. Der Marker "TRAILER" hat ab sofort IMAGE-Eigenschaften wie z.B. .file etc.
- In Zeile 14 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 16 und 17 geben wir die grundlegenden Eigenschaften Abmaße und Hintergrundfarbe der dynamischen Grafik an.

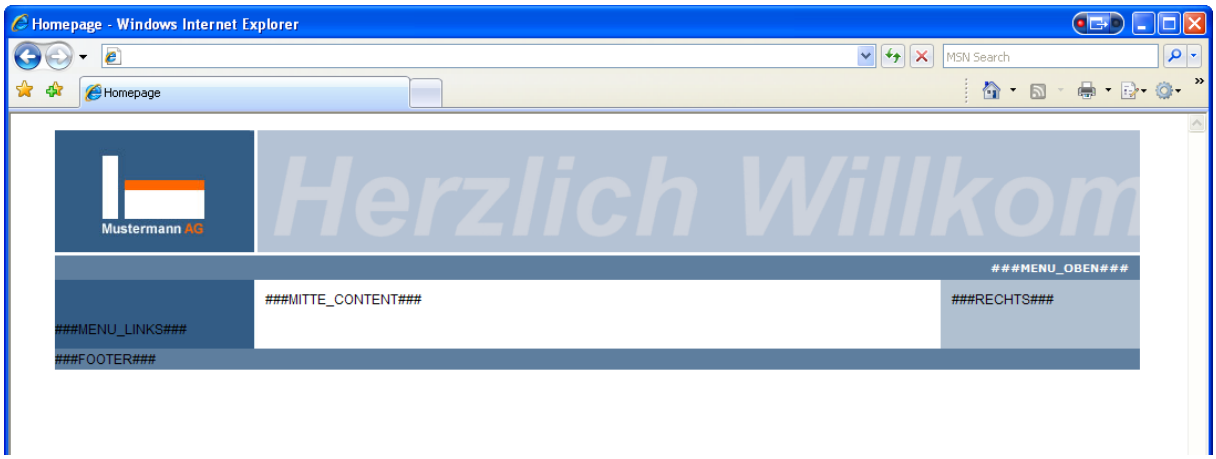
5.4.7 Text auf den Trailer rendern

Unser oben erstellter Trailer ist zurzeit lediglich eine hellblaue 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 TYPO3 Text-CObject. 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 {
       [...]
08     10 = TEMPLATE
09     10.template = FILE
10     10.template.file = fileadmin/vorlage.html
11     10.workOnSubpart = DOKUMENT
12     10.marks{
13           TRAILER = IMAGE
14           TRAILER.file = GIFBUILDER
15           TRAILER.file{
16                XY = 797, 110
17                backColor = #B0C0D0
18                10 = TEXT
19                10.text = Herzlich Willkommen
20                10.fontSize = 100
21                10.fontFile = fileadmin/fonts/arialbi.ttf
22                10.fontColor = #C8D3DE
23                10.niceText = 1
24                10.offset = 10,95
25           }
26     }
27 }
```



- In Zeile 18 wird auf der grafischen Ebene 10 eine grafische TEXT-Instanz erzeugt. Diese Ebenen sind vom Verständnis her mit den Photoshop-Ebenen zu vergleichen – mehrere

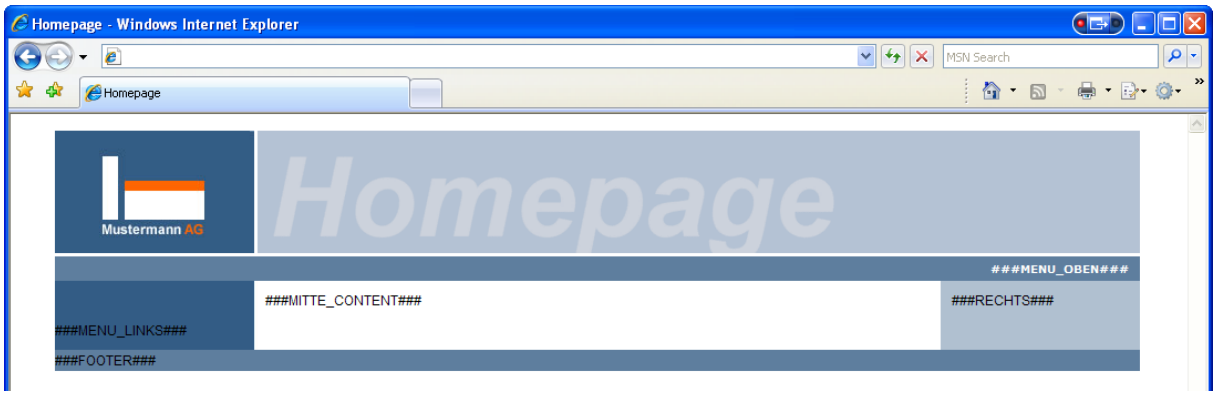
Elemente können übereinander liegen. In unserem Fall liegt die Ebene 10 somit auf der Hintergrundebene.

- In Zeile 19 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 20 bis 22 wird den Eigenschaften fontFile, fontSize und fontColor jeweils ein Wert zugewiesen. Über diese Eigenschaften wird die Formatierung des Textes angegeben.
- In Zeile 23 wird durch die Eigenschaft "niceText" der Kantenglätter aktiviert.
- In Zeile 24 werden die Koordination angegeben, an deren Position 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 797 Pixel breit und 110 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.

5.4.8 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 Seitentitel lautet "title". Wir ändern unser Template wie folgt:

```
18              10 = TEXT
19              10.text.field = title
20              10.fontSize = 100
```

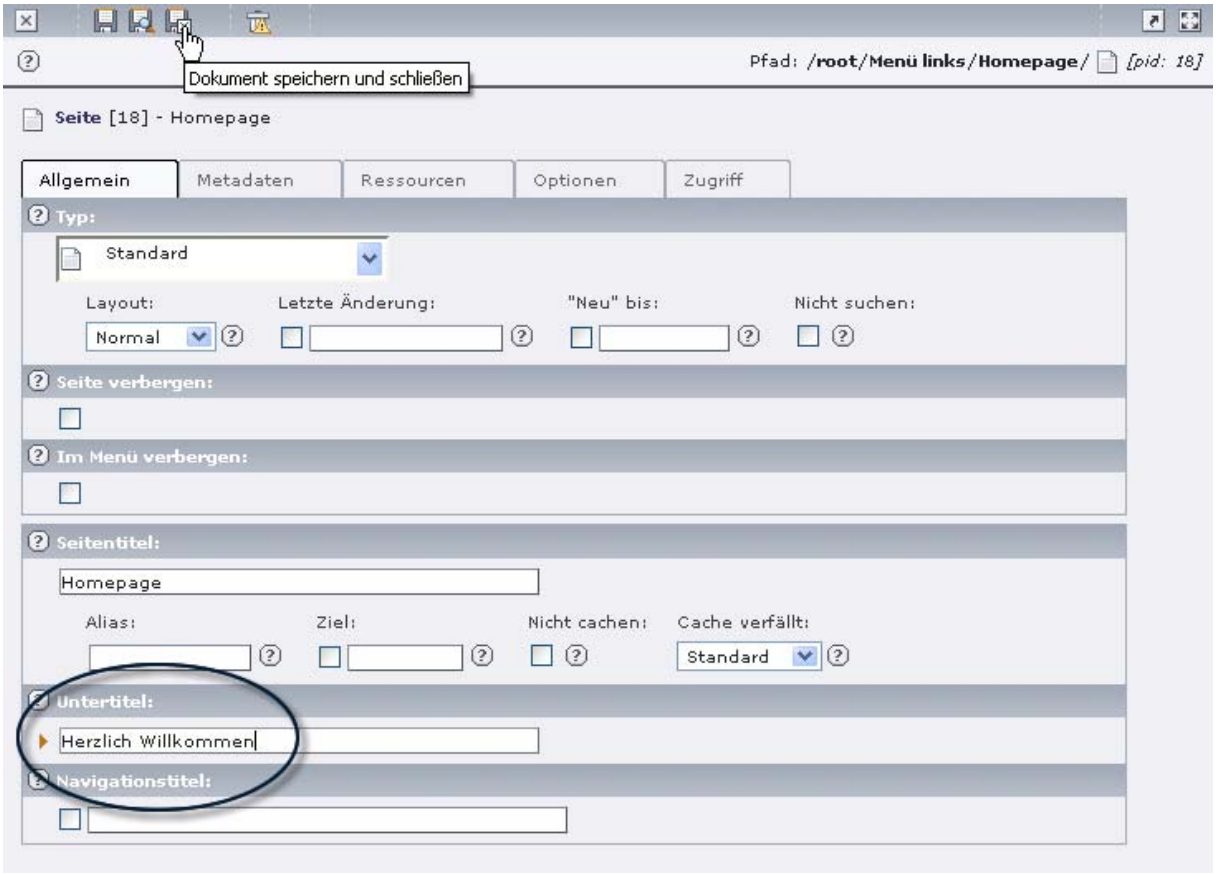


Nun soll aber auf unserer Homepage im Trailer nicht "Homepage", sondern "Herzlich Willkommen" stehen. Wir müssen es nun also ermöglichen, dass statt dem Seitentitel ein anderes Datenbankfeld ausgelesen wird. In Kapitel 4.4.1 wurde hierzu die Möglichkeit vorgestellt, mittels **//** mehrere Datenbankfelder 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:

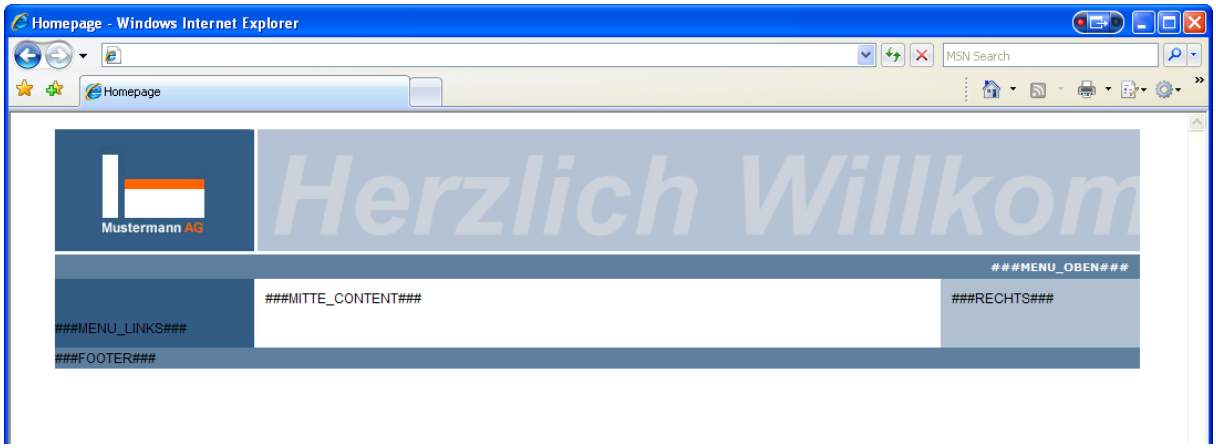
Auf der rechten Seite geben wir dann auf dem Reiter „Allgemein“ im Feld "Untertitel" unsere Trailer-Überschrift "Herzlich Willkommen" an und speichern unsere Seite.



In unserem Template müssen wir folgende Änderung vornehmen:

```
18              10 = TEXT
19              10.text.field = subtitle // title
20              10.fontSize = 100
```

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



5.4.9 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:

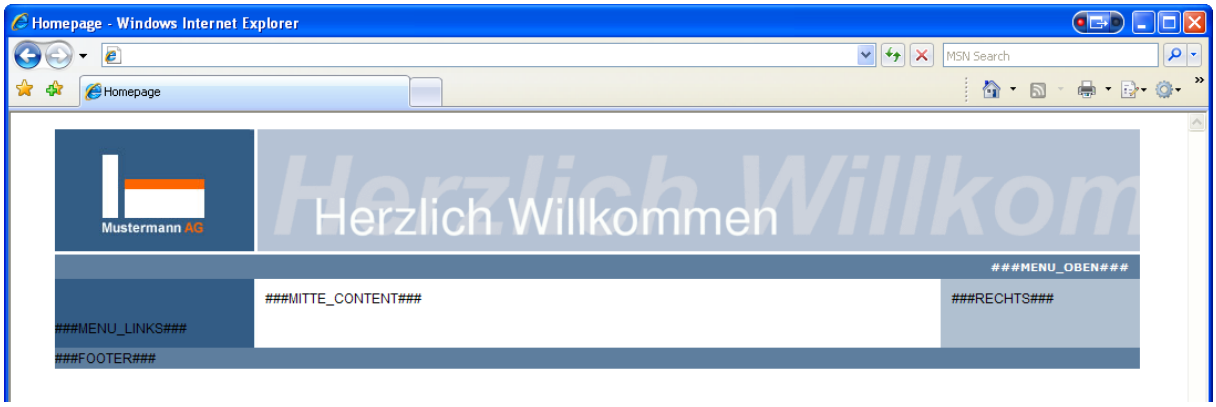
- Schrift: Arial Bold (fileadmin/fonts/arailb.ttf)
- Schriftgröße: 45 Punkt
- Schriftfarbe: Weiß (#FFFFFF)

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

Beispiel 5

```
01 seite = PAGE
02 seite {
       [...]
08     10 = TEMPLATE
09     10.template = FILE
10     10.template.file = fileadmin/vorlage.html
11     10.workOnSubpart = DOKUMENT
12     10.marks{
13          TRAILER = IMAGE
14          TRAILER.file = GIFBUILDER
```

```
15          TRAILER.file{
16               XY = 797, 110
17               backColor = #B0C0D0
18               10 = TEXT
19               10.text.field = subtitle // title
20               10.fontSize = 100
21               10.fontFile = fileadmin/fonts/arialbi.ttf
22               10.fontColor = #C8D3DE
23               10.niceText = 1
24               10.offset = 10,95
25               20 = TEXT
26               20.text.field = subtitle // title
27               20.fontSize = 45
28               20.fontFile = fileadmin/fonts/arialb.ttf
29               20.fontColor = #FFFFFF
30               20.niceText = 1
31               20.offset = 50,95
32          }
33     }
34 }
```



5.5 TMENU: Menü oben erstellen

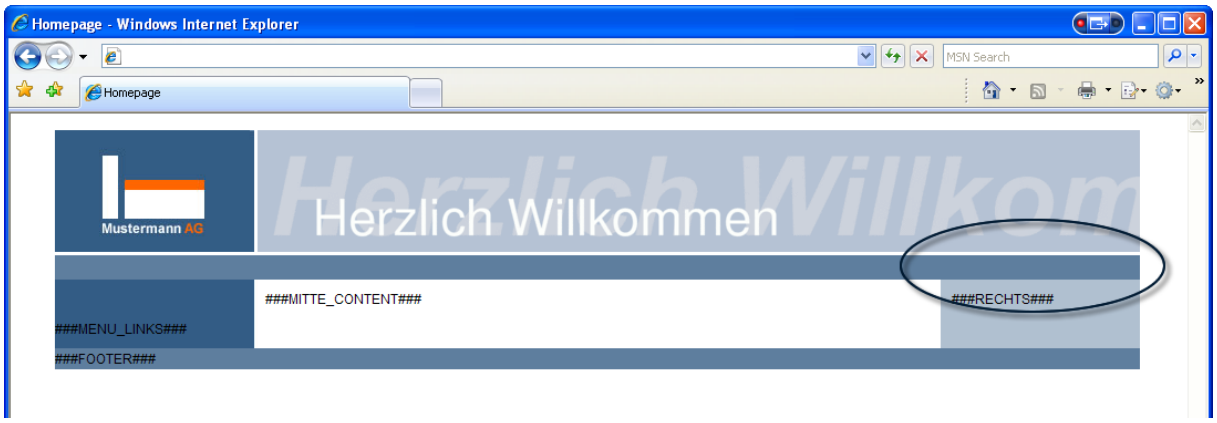
Nachdem wir nun unseren Trailer fertig gestellt haben, können wir uns unserem ersten Menü im oberen Bereich widmen. Dieses Menü soll ein reines Textmenü werden – die einzelnen Menüelemente werden aus unserer Seitenstruktur entnommen.

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

```
01 seite = PAGE
02 seite {
       [...]
08     10 = TEMPLATE
09     10.template = FILE
10     10.template.file = fileadmin/vorlage.html
11     10.workOnSubpart = DOKUMENT
```

```
12     10.marks{
13          TRAILER = IMAGE
14          TRAILER.file = GIFBUILDER
15          TRAILER.file{
16               XY = 797, 110
17               [...]
31               20.offset = 50,95
32          }
33          MENU_OBEN = HMENU
34     }
35 }
```

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.



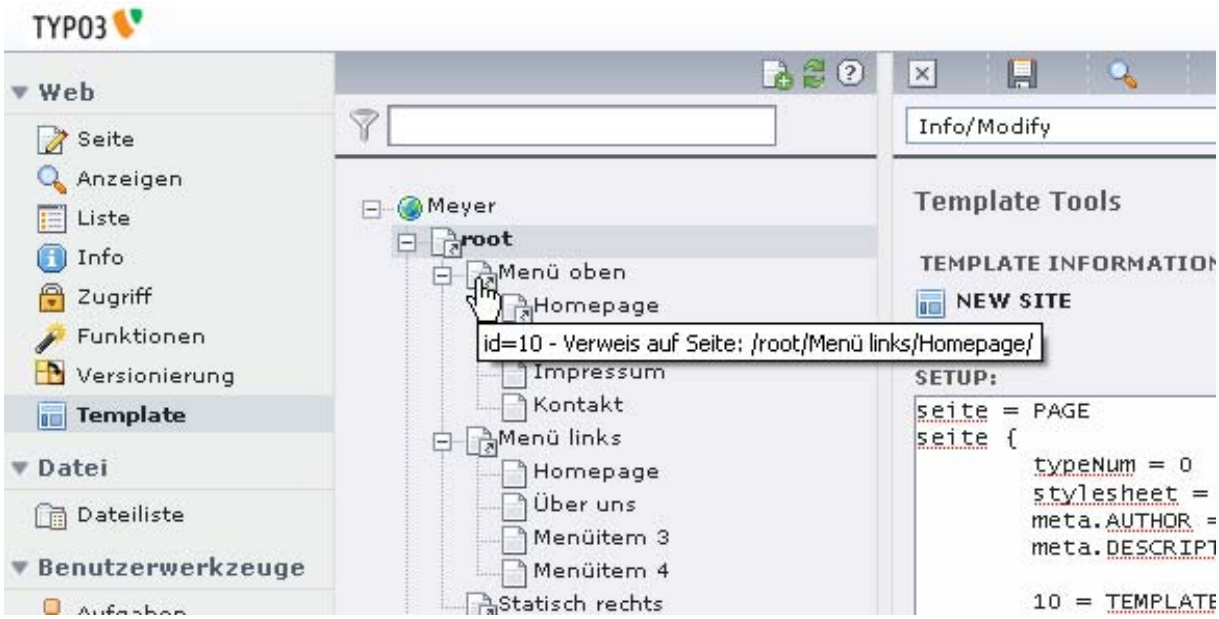
Wenn der Marker nicht angesprochen wurde, überprüfen Sie bitte folgende Punkte:

- Zunächst, wie im Kapitel 5.4.5 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 Klammern "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 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 {
       [...]
12     10.marks{
13           TRAILER = IMAGE
14           TRAILER.file = GIFBUILDER
15           TRAILER.file{
17                 [...]
32           }
33           MENU_OBEN = HMENU
34           MENU_OBEN.special = directory
35           MENU_OBEN.special.value = 10
36     }
37 }
```

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.

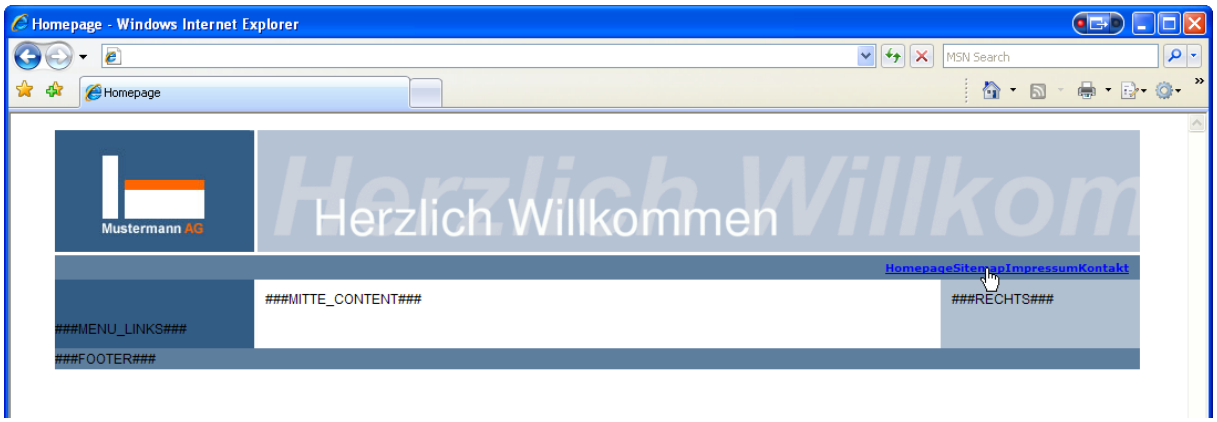
- In Zeile 33 erzeugen wir auf dem Marker "MENU_OBEN" eine Instanz des HMENUObjektes
- In Zeile 34 geben wir über die special-Eigenschaft an, dass es sich bei dem erzeugten Menü um ein "diretory-Menü" handeln soll, also um ein Menü aus einem Verzeichnis. Weitere Menü-Möglichkeiten (wie z.B. vor/zurück, Klickpfade etc. im Kapitel 4.12.2)
- In Zeile 35 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!

5.5.1 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 {
       [...]
12     10.marks{
13           TRAILER = IMAGE
14           TRAILER.file = GIFBUILDER
15           TRAILER.file{
17                [...]
32           }
33           MENU_OBEN = HMENU
34           MENU_OBEN.special = directory
35           MENU_OBEN.special.value = 10
36           MENU_OBEN.1 = TMENU
37           MENU_OBEN.1.NO = 1
38     }
39 }
```

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

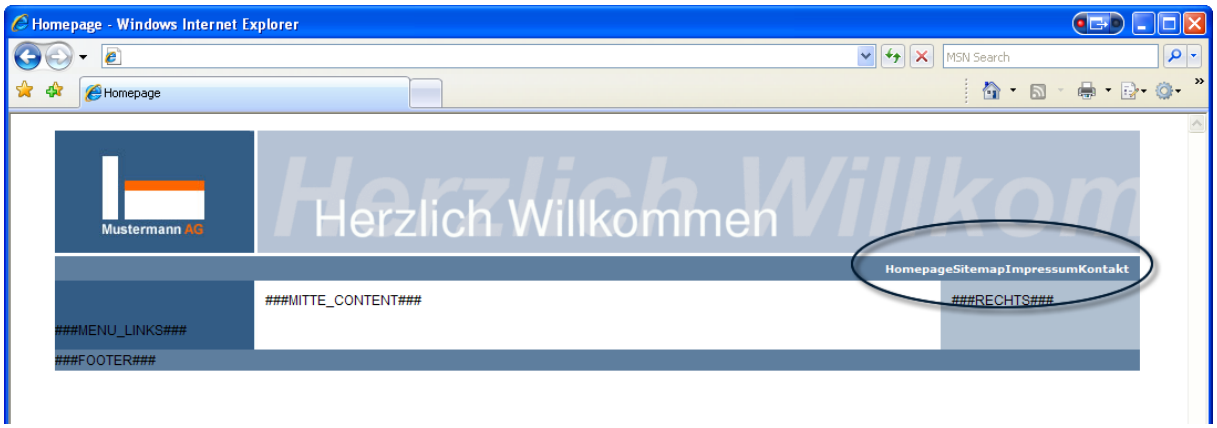


- In Zeile 36 geben wir an, dass auf der ersten Menüebene ein Textmenü (TMENU) verwendet werden soll. Es wird somit auf der ersten Ebene eine Instanz des TMENUs erzeugt und es stehen alle Eigenschaften des TMENUs zur Verfügung.
- In Zeile 37 geben wir mittels "1.NO = 1" an, dass wir den normalen Zustand aktivieren wollen.

5.5.2 Stylesheet im Menü 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 [...]
38 MENU_OBEN.1 = TMENU
39 MENU_OBEN.1.NO = 1
40 MENU_OBEN.1.NO.ATagParams = class="linkWeiss"
```

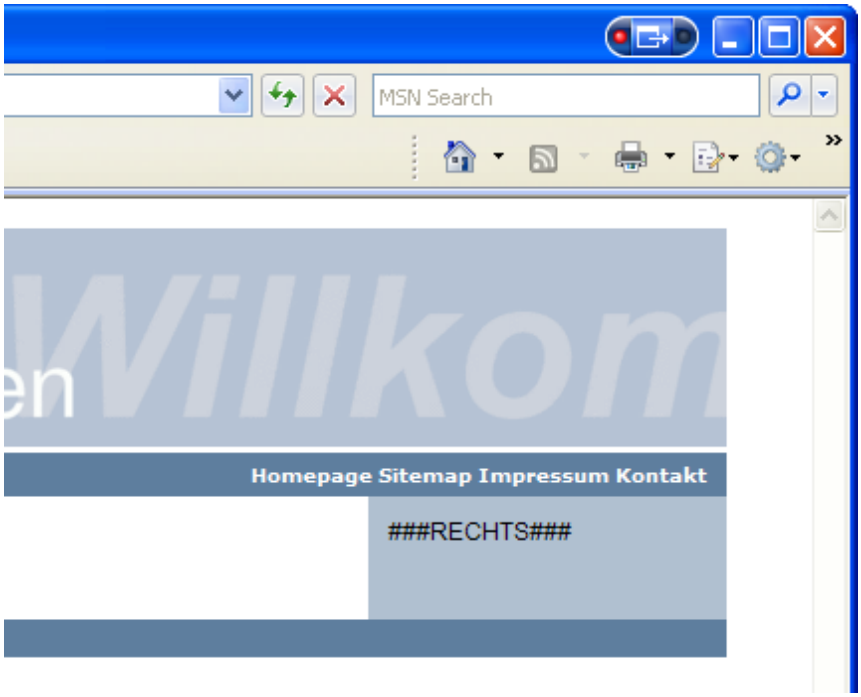


5.5.3 Menüeinträge voneinander trennen

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

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

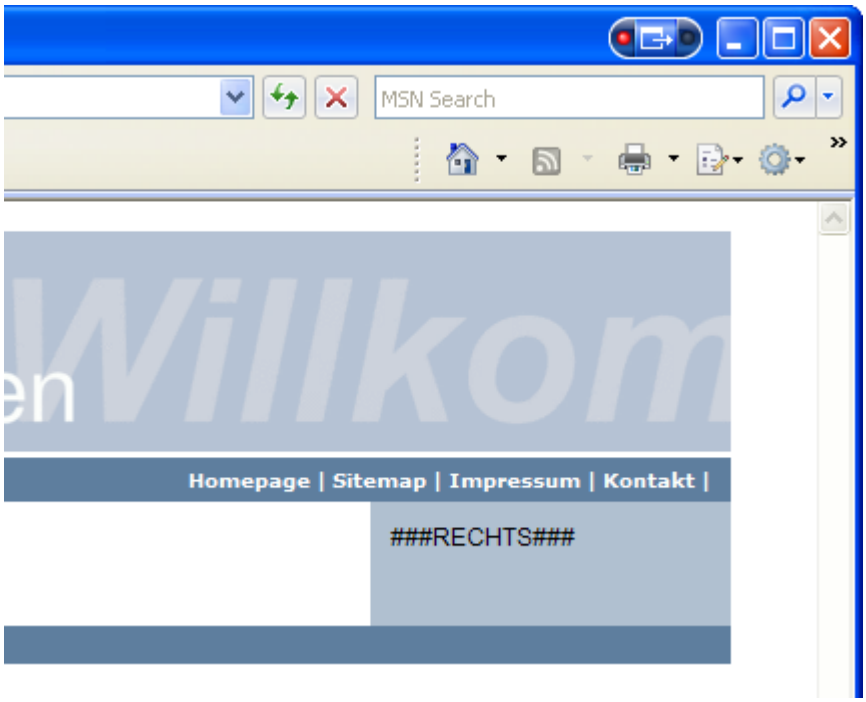
Das Ergebnis kann uns zufrieden stellen, jedoch hat der Grafiker uns eine andere Vorgabe gemacht: Zwischen jedem Menüeintrag soll ein Strich angezeigt werden.



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:

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

Beim linkWrap wird nun zunächst ein Leerzeichen ausgegeben, dann folgt der Menüeintrag incl. dem A-Tag, erneut ein 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 zu viel haben:



5.5.4 optionSplit für Text-Menüs mit Pipe-Symbol

Eine Lösung ist nur mittels der optionSplit-Möglichkeit gegeben. Im Kapitel 4.14.2 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 vor:

- 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 "` |`".

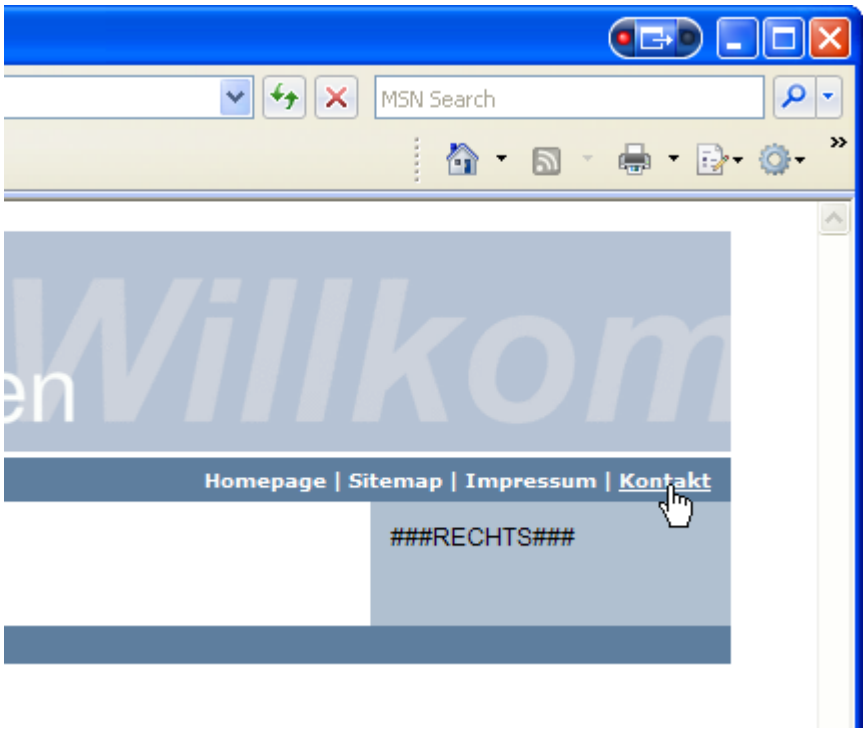
Daraus folgt der (durchaus umfangreiche) linkWrap:

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

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



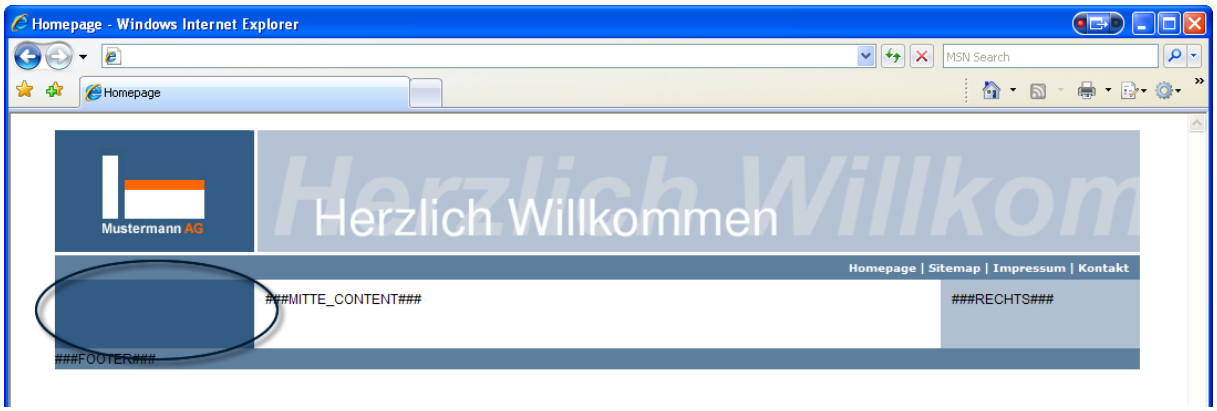
5.6 GMENU: Das linke Menü erstellen

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    stylesheet = fileadmin/style.css
05    meta.AUTHOR = Robert Meyer
06    meta.DESCRIPTION = Hier steht eine Beschreibung
07    10 = TEMPLATE
08    10.template = FILE
09    10.template.file = fileadmin/vorlage.html
10    10.workOnSubpart = DOKUMENT
11
12    10.marks{
13          TRAILER = IMAGE
14          TRAILER.file = GIFBUILDER
15          TRAILER.file{
16                XY = 797, 110
17                backColor = #B0C0D0
18                10 = TEXT
19                10.text.field = subtitle // title
20                10.fontSize = 100
21                10.fontFile = fileadmin/fonts/arialbi.ttf
22                10.fontColor = #C8D3DE
23                10.niceText = 1
24                10.offset = 10,95
25
26                20 = TEXT
27                20.text.field = subtitle // title
28                20.fontSize = 45
29                20.fontFile = fileadmin/fonts/arialb.ttf
30                20.fontColor = #FFFFFF
31                20.niceText = 1
32                20.offset = 50,95
33          }
34
35          MENU_OBEN = HMENU
36          MENU_OBEN.special = directory
37          MENU_OBEN.special.value = 10
38          MENU_OBEN.1 = TMENU
39          MENU_OBEN.1.NO = 1
40          MENU_OBEN.1.NO.ATagParams = class="linkWeiss"
41          MENU_OBEN.1.NO.linkWrap =   | &#124; |*|
             | &#124; |*|  |
42
43          MENU_LINKS = HMENU
44          MENU_LINKS.special = directory
45          MENU_LINKS.special.value = 9
46    }
47 }
```



- In Zeile 43 wird auf dem Marker "MENU_LINKS" eine HMENU-Instanz erzeugt. Der Marker MENU_LINKS hat somit generelle Menü-Eigenschaften
- In Zeile 44 teilen wir TYPO3 mit, dass das Menü ein Menü spezial Eigenschaft „directory“ sein soll, welches von einer bestimmten Seite aufgebaut werden soll
- In Zeile 45 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!

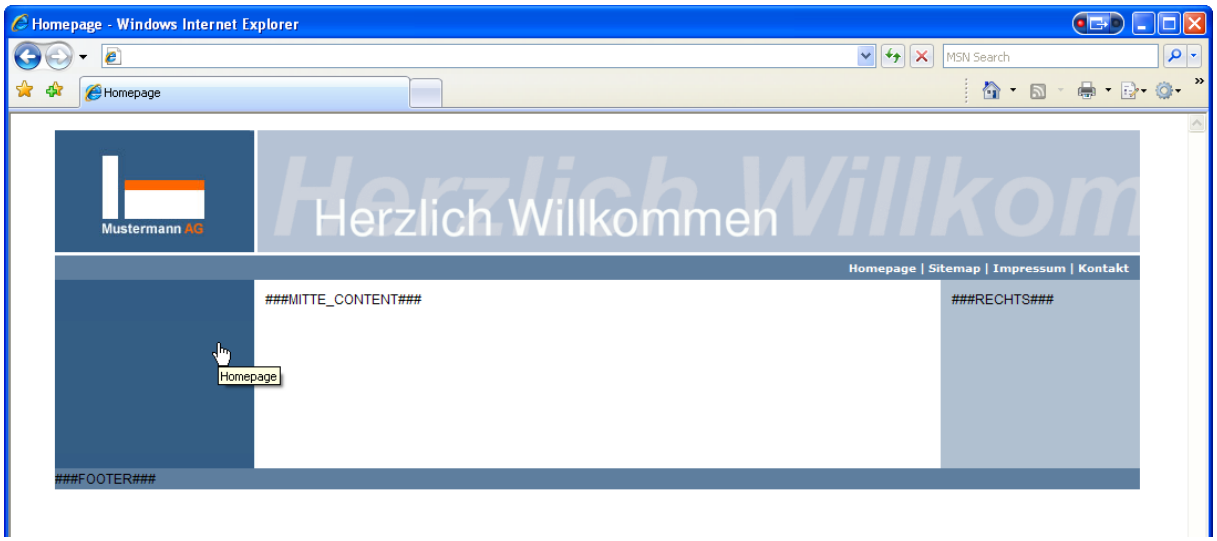
Ein Betrachten der Seite im Frontend wird kein Menü darstellen lassen – der Marker selbst sollte aber nicht mehr dargestellt werden. Falls der Marker noch dargestellt wird, finden Sie entsprechende Informationen im Kapitel 5.4.5.

5.6.1 Grafische Menüeinträge erzeugen lassen

Jeder Menüeintrag soll mit einer dunkelblauen 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:

```
43          MENU_LINKS = HMENU
44          MENU_LINKS.special = directory
45          MENU_LINKS.special.value = 9
46          MENU_LINKS.1 = GMENU
47          MENU_LINKS.1.NO = 1
48          MENU_LINKS.1.NO {
49                XY = 180,25
50                backColor = #335C85
51          }
```



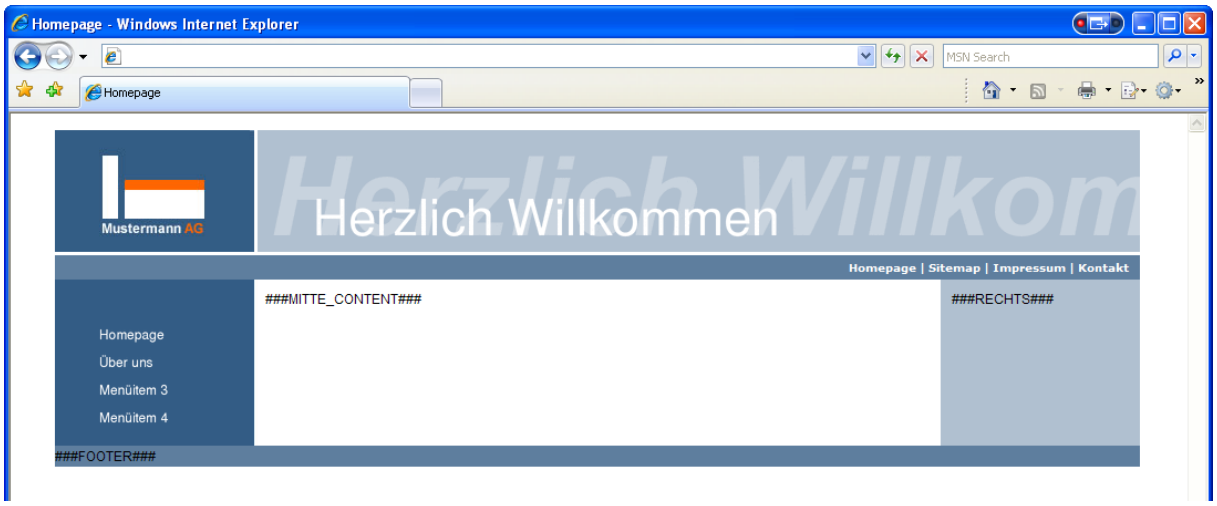
Betrachten wir das Ergebnis im Frontend, können wir erkennen, dass sich bereits etwas getan hat. Einzelne Menüpunkte sind zwar nicht direkt sichtbar – ein Mouse-Over über die grüne Fläche zeigt jedoch schon, dass sich hier Menüelemente befinden.

5.6.2 Text auf die Grafik rendern

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

```
43          MENU_LINKS = HMENU
44          MENU_LINKS.special = directory
45          MENU_LINKS.special.value = 9
46          MENU_LINKS.1 = GMENU
47          MENU_LINKS.1.NO = 1
48          MENU_LINKS.1.NO {
49                XY = 180,25
50                backColor = #335C85
51
52                10 = TEXT
53                10.text.field = title
54                10.fontColor = #FFFFFF
55                10.fontFile = fileadmin/fonts/arialb.ttf
56                10.fontSize = 12
57                10.niceText = 1
58                10.offset = 40,17
59          }
```

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



- In Zeile 52 wird als Ebene 10 eine Textebene angelegt. Diese Ebene 10 befindet sich auf der Hintergrundebene.
- In Zeile 53 wird angegeben, woher der Inhalt genommen werden soll. Hier: dynamisch aus dem Datenbankfeld "title".
- In den Zeilen 54 bis 56 werden die Eigenschaften zur verwendeten Schrift angegeben: Schriftfarbe, Schriftdatei und Schriftgröße (in Punkt).
- In Zeile 57 wird für die Schrift der Kantenglätter aktiviert. Hierdurch werden runde Kanten ermöglicht – der Text wirkt nicht mehr "pixelig".
- In Zeile 58 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.

Die Grafiken werden als Standard-Einstellung als GIF Dateien erzeugt. TYPO3 benötigt zur Grafik Erzeugung bestimmte PHP Bibliotheken – GDlib. Bei der Verwendung der GDlib-Version 2 kann es bei den GIF-Grafiken zu Abweichungen in der Darstellung der Farben kommen. Wechseln Sie im TYPO3 Install Tool auf eine Grafikerzeugung von PNG Dateien über die Einstellung - **$TYPO3_CONF_VARS['GFX']['gdlib_png'] = '1';**.

5.6.3 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:

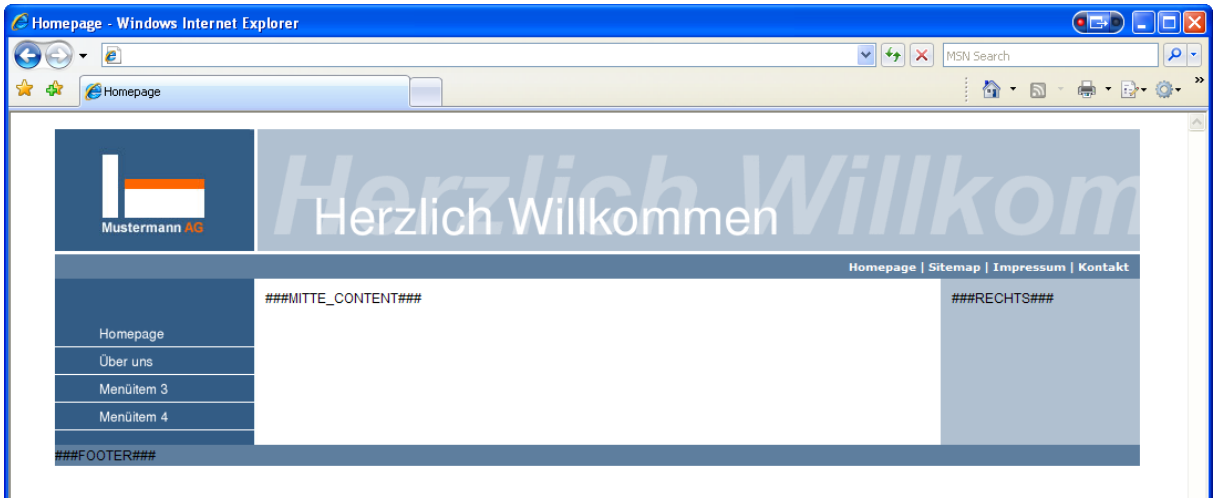
```
wrap = |<img src="fileadmin/linie.gif" width="146" height="1">
```

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 in die Ebene 20 eine Linie laden. Diese Linie ist aber nicht vorhanden und soll dynamisch mittels des Gifbuilders erzeugt werden:

```
43          MENU_LINKS = HMENU
44          MENU_LINKS.special = directory
45          MENU_LINKS.special.value = 9
46          MENU_LINKS.1 = GMENU
47          MENU_LINKS.1.NO = 1
48          MENU_LINKS.1.NO {
49                XY = 180,25
50                backColor = #335C85
51
52                10 = TEXT
53                10.text.field = title
54                10.fontColor = #FFFFFF
55                10.fontFile = fileadmin/fonts/arialb.ttf
56                10.fontSize = 12
57                10.niceText = 1
58                10.offset = 40,17
59
60                20 = IMAGE
61                20.file = GIFBUILDER
62                20.file{
63                      XY = 180,1
64                      backColor = #FFFFFF
65                }
66                20.offset = 0,24
67          }
```



- In Zeile 60 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 61 geben wir an, dass die Grafik dynamisch mit dem GIFBUILDER erzeugt werden soll.
- In den Zeilen 63 und 64 geben wir an, dass diese Grafik 180 Pixel lang sein soll und nur 1 Pixel hoch. Als Hintergrundfarbe wird weiß angegeben.
- In Zeile 66 verschieben wir die erzeugte Grafik um 0 Pixel nach rechts und 24 Pixel nach unten.

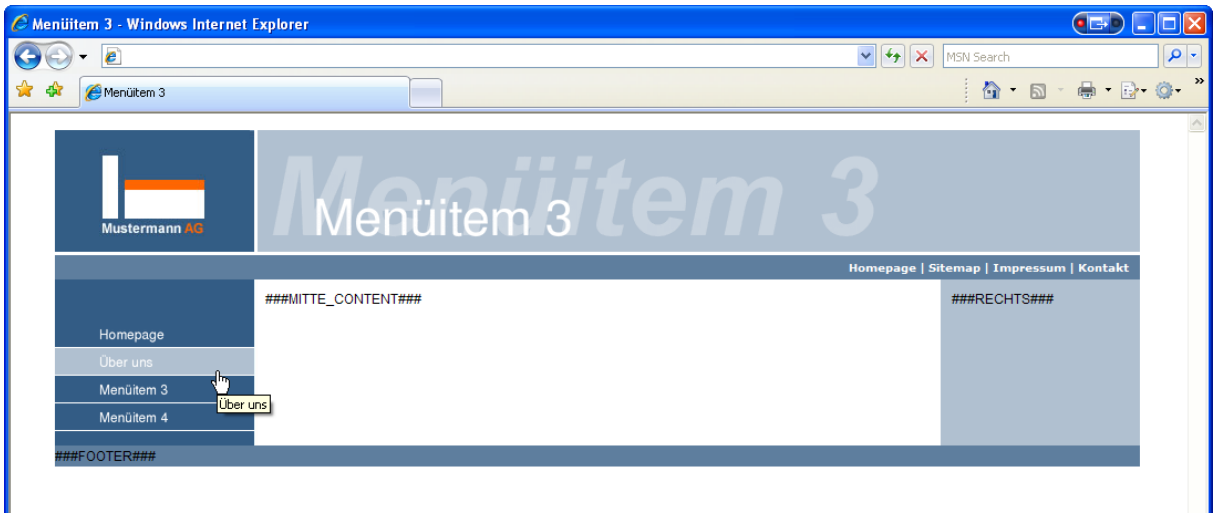
5.6.4 Unterschiedliche Menüzustände integrieren

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-Blauton als Schatten des Textes erscheinen.

Richten wir hierzu zunächst den RollOver mit veränderter Hintergrundfarbe ein:

```
43          MENU_LINKS = HMENU
            [...]
46          MENU_LINKS.1 = GMENU
47          MENU_LINKS.1.NO = 1
48          MENU_LINKS.1.NO {
51                [...]
52                10 = TEXT
                  [...]
60                20 = IMAGE
                  [...]
67          }
68          MENU_LINKS.1.RO < .MENU_LINKS.1.NO
69          MENU_LINKS.1.RO.backColor = #B0C0D0
```

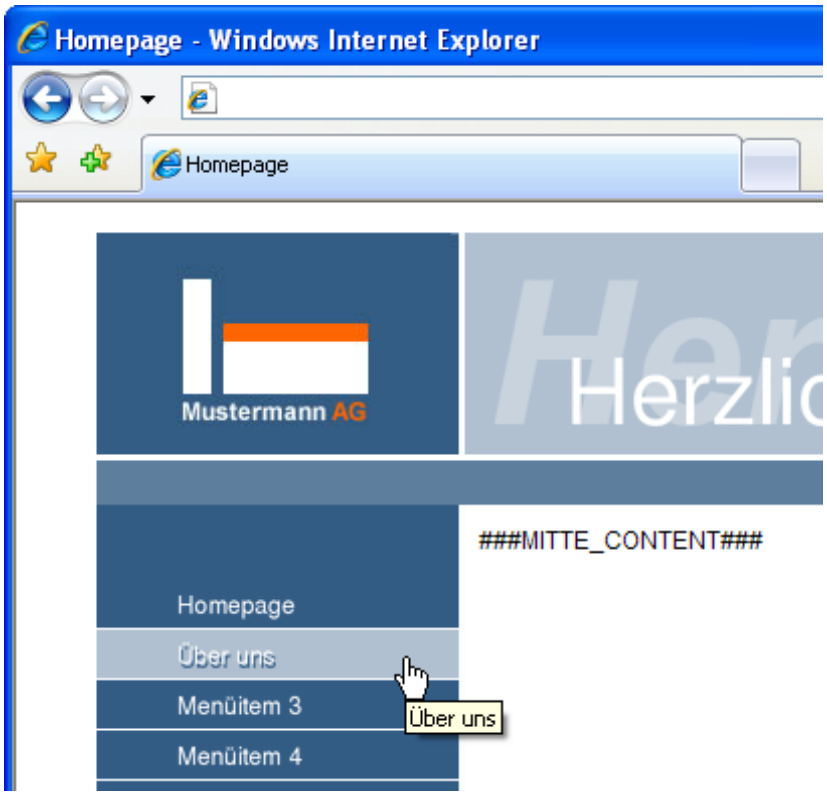


- In Zeile 68 wird mittels relativer Kopie der Zustand NO in den Zustand RO kopiert. Bei diesem Kopiervorgang werden sämtliche Eigenschaften mit kopiert und stehen als Kopie in RO zur Verfügung. Da in Zeile 47 der Zustand NO aktiviert wurde, wird durch den Kopiervorgang nun der Zustand RO ebenfalls aktiviert.
- In Zeile 69 wurde die Eigenschaft backColor, die durch den Kopiervorgang mit dem dunkleren Blauton zugewiesen wurde, überschrieben.

5.6.5 Eine zusätzliche Textebene hinzufügen

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

```
43          MENU_LINKS = HMENU
            [...]
46          MENU_LINKS.1 = GMENU
47          MENU_LINKS.1.NO = 1
48          MENU_LINKS.1.NO {
51                [...]
52                10 = TEXT
                  [...]
60                20 = IMAGE
                  [...]
67          }
68          MENU_LINKS.1.RO < .MENU_LINKS.1.NO
69          MENU_LINKS.1.RO {
70                tmp < .backColor
71                backColor = #B0C0D0
72                5 < .10
73                5.fontColor < .tmp
74                5.offset = 42,19
75          }
```



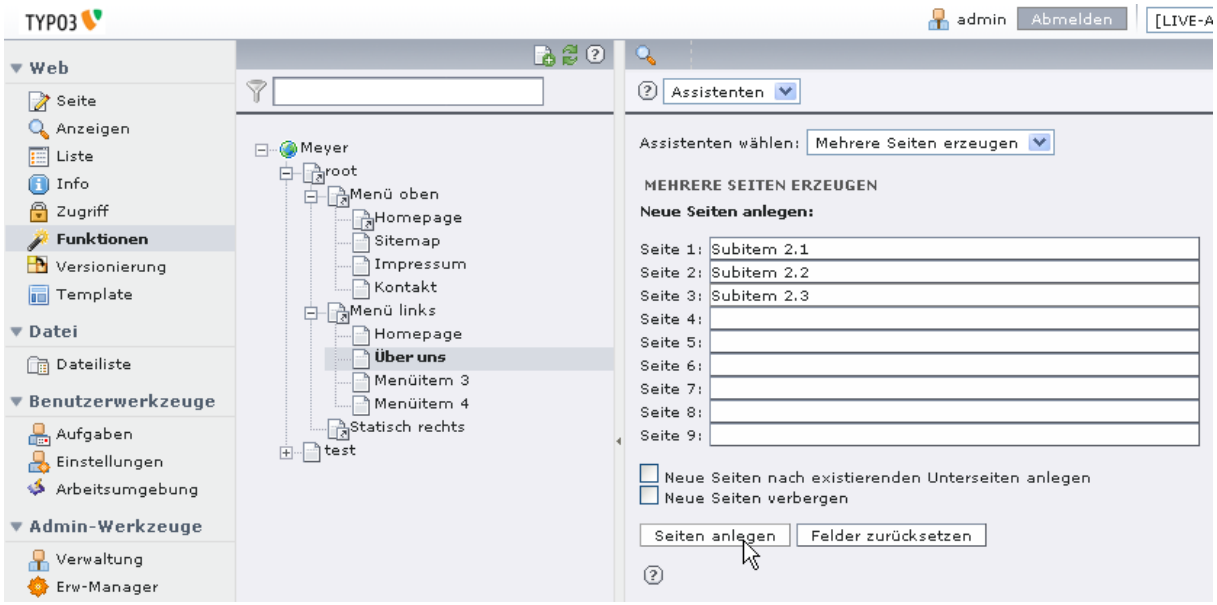
- In Zeile 69 haben wir zur Vereinfachung ausgeklammert. Die Klammerung endet in Zeile 75.
- In Zeile 70 wurde die bisherige Hintergrundfarbe "zwischengespeichert". Die Variable "tmp" ist frei gewählt – auch hätte hier als Variable "meyer" verwendet werden können. Die ursprüngliche Hintergrundfarbe wird später für die Textfarbe benötigt.
- In Zeile 71 wird die Hintergrundfarbe mit einem helleren Blauton überschrieben
- In Zeile 72 wird die gesamte Textebene 10 in die Textebene 5 kopiert.
- Die neueTextebene 5 liegt unterhalb der Ebene 10. Die Kopie findet hier Anwendung, da fast alle Eigenschaften des Textes übernommen werden sollen – lediglich der Offset und die Textfarbe werden sich ändern.
- In Zeile 73 wird die zwischengespeicherte Farbe in die Eigenschaft "fontColor" kopiert. Somit hat fontColor jetzt die gleiche Farbe wie die ursprüngliche Hintergrundfarbe.
- In Zeile 74 findet die Verschiebung statt – zwei Pixel weiter rechts und zwei Pixel weiter unten als der Text, der auf der Ebene 10 dargestellt wird.

5.6.6 Eine zweite Menüebene und weitere Zustände hinzufügen

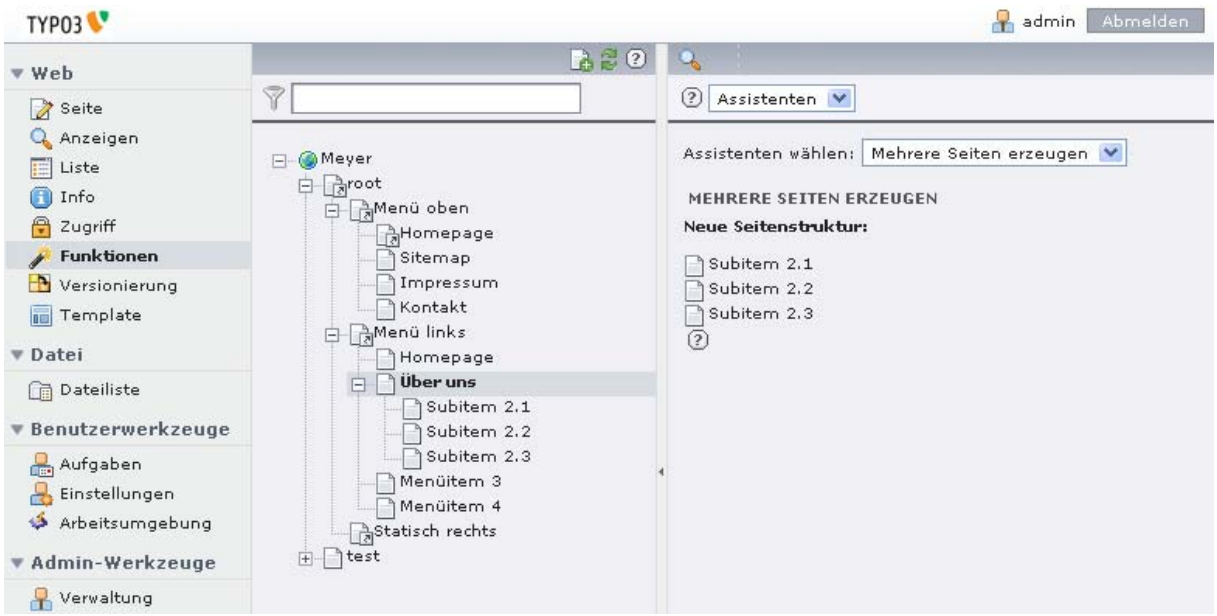
Wir können selbstverständlich auch weitere Menüebenen zu unserem Menü auf der linken Seite hinzufügen. Der Grafiker hat hierzu keine grafischen Vorgaben gemacht. Wir machen uns die Regeln somit selber: Die Menüpunkte sollen eine hellgrüne Hintergrundfarbe haben (wie in der rechten Spalte), der Text soll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

5.6.7 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:



5.6.8 Template erweitern

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

```
43          MENU_LINKS = HMENU
            [...]
```

```
46          MENU_LINKS.1 = GMENU
47          MENU_LINKS.1.NO = 1
48          MENU_LINKS.1.NO {
51                [...]
52                10 = TEXT
                  [...]
60                20 = IMAGE
                  [...]
67          }
68          MENU_LINKS.1.RO < .MENU_LINKS.1.NO
69          MENU_LINKS.1.RO {
                  [...]
75          }
76          MENU_LINKS.2 < .MENU_LINKS.1
```

- In Zeile 76 wird auf die zweite Navigationsebene 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 öffnet eine zweite Menüebene für die Seite "Über uns":

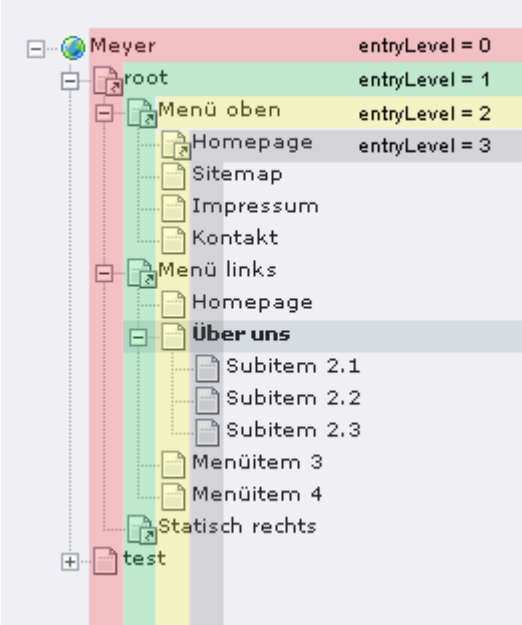
5.6.9 entryLevel für weitere Menüebenen (für TYPO3 Versionen kleiner 4.2)

Sollten Sie im Frontend keine Ausgabe der Unterseiten in der Navigation erhalten, nutzen Sie vermutlich eine TYPO3-Version kleiner als 4.2.

In diesen Versionen müsste bei der Verwendung der special-Eigenschaft des HMENU-Objektes eine Eigenschaft "entryLevel" gesetzt werden. 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 TypoScript haben wir durch "special.value = 9" angegeben, dass das Menü von der Seite 9 ("Menü links") 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
```

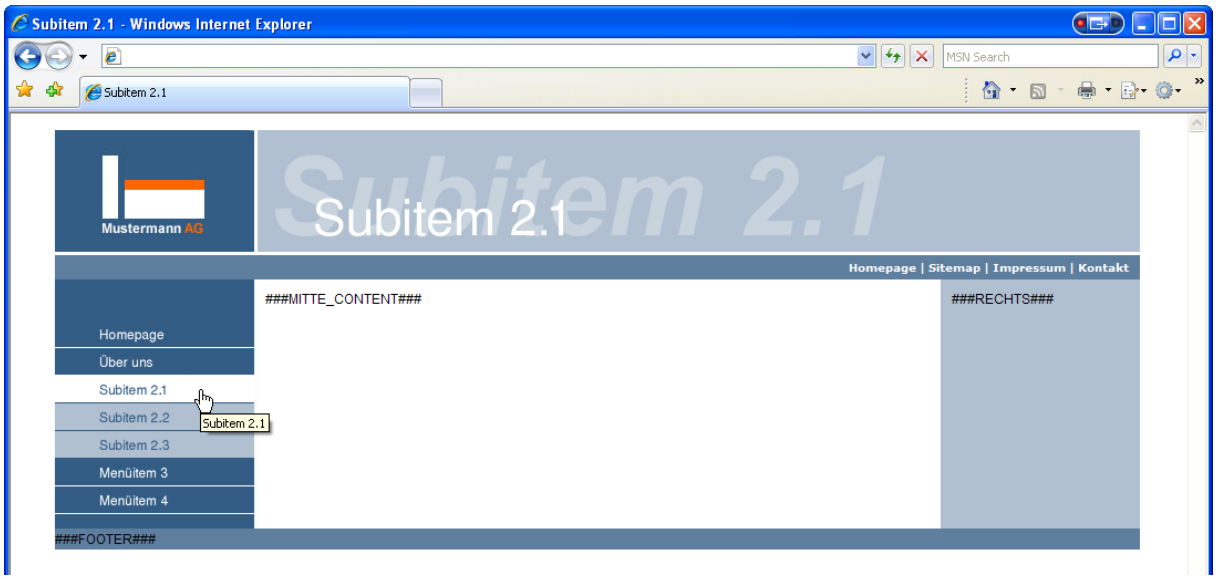
Ein erneutes Betrachten unserer Seite im Frontend sollte dann bei älteren TYPO3-Versionen zufrieden stimmen.

5.6.10 Die zweite Ebene optisch anpassen

Für die zweite Menüebene sollten nun aber einige Eigenschaften anders sein: hellblauer Hintergrund, dunkelblaue, kleinere und eingerückte Schrift:

```
43          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 {
                [...]
76          }
77          MENU_LINKS.2 < .MENU_LINKS.1
78          MENU_LINKS.2 {
79              NO.backColor = #B0C0D0
80              NO.10.fontColor = #335C85
81              NO.20.file.backColor = #335C85
82              RO < .NO
83              RO.backColor = #FFFFFF
84              RO.5 >
85          }
```



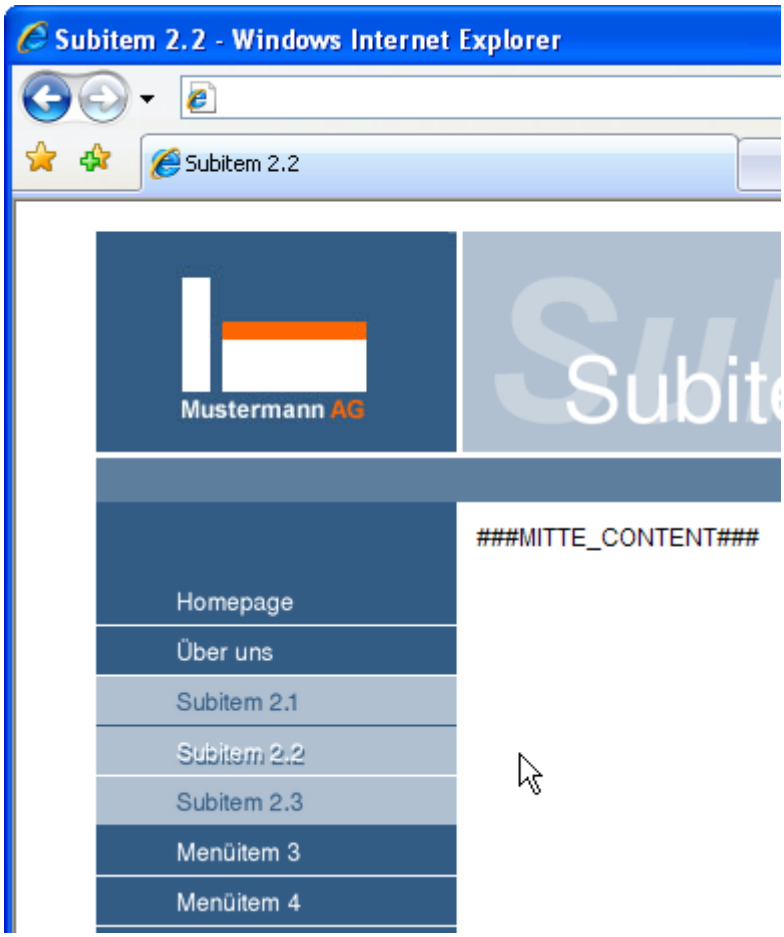
- In Zeile 77 werden die Eigenschaften der ersten Ebene in die zweite Ebene kopiert. Die zweite Ebene ist damit funktionsfähig.
- In Zeile 79 wird die Eigenschaft "backColor" für den normalen Zustand überschrieben und auf ein helles Blau gesetzt.
- In Zeile 80 wird für die Textebene 10 die Schriftfarbe auf ein dunkles Blau gesetzt.
- In Zeile 81 wird die Farbe für die Linie auf ein dunkles Blau gesetzt.
- In Zeile 82 werden die Eigenschaften des normalen Zustandes 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 Hintergrundfarbe 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.

5.6.11 Den Menü-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:

```
43          MENU_LINKS = HMENU
            [...]
47          MENU_LINKS.1 = GMENU
48          MENU_LINKS.1.NO = 1
49          MENU_LINKS.1.NO {
                  [...]
68          }
69          MENU_LINKS.1.RO < .MENU_LINKS.1.NO
                  [...]
77          MENU_LINKS.1.CUR < .MENU_LINKS.1.RO
78          MENU_LINKS.2 < .MENU_LINKS.1
79          MENU_LINKS.2 {
```



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 derjenige 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 {NO.backColor = #B0C0D0
80                NO.10.fontColor = #335C85
81                NO.20.file.backColor = #335C85
82                RO < .NO
83                RO.backColor = #FFFFFF
84                RO.5 >
85                CUR < .RO
86          }
```

5.6.12 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, das nebeneinander stehen soll?

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

```
[...]
<a href="18.0.html" ….> <img src="typo3temp/…> </a>
<a href="19.0.html" ….> <img src="typo3temp/…> </a>
<a href="20.0.html" ….> <img src="typo3temp/…> </a>
[...]
```

Der Zeilenumbruch wird (freundlicherweise) vom Browser eigenmächtig vorgenommen, bedingt durch den Container, dem wir eine Breite von 180 Pixeln in der Designvorlage zugewiesen haben.

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

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

```
47          MENU_LINKS.1.NO = 1
48          MENU_LINKS.1.NO {
49                wrap = |<br>
```

5.6.13 Das gesamte TypoScript des linken Menüs:

```
01    page = PAGE
      [...]
43          MENU_LINKS = HMENU
44          MENU_LINKS.special = directory
45          MENU_LINKS.special.value = 9
46          MENU_LINKS.entryLevel = 1
47          MENU_LINKS.1 = GMENU
48          MENU_LINKS.1.NO = 1
49          MENU_LINKS.1.NO {
50                wrap = |<br>
51                XY = 180,25
52                backColor = #335C85
53
54                10 = TEXT
55                10.text.field = title
56                10.fontColor = #FFFFFF
57                10.fontFile = fileadmin/fonts/arialb.ttf
58                10.fontSize = 12
59                10.niceText = 1
60                10.offset = 40,17
61
62                20 = IMAGE
63                20.file = GIFBUILDER
64                20.file{
65                      XY = 180,1
66                      backColor = #FFFFFF
67                }
68                20.offset = 0,24
69          }
70          MENU_LINKS.1.RO < .MENU_LINKS.1.NO
71          MENU_LINKS.1.RO {
72                tmp < .backColor
73                backColor = #B0C0D0
74                5 < .10
75                5.fontColor < .tmp
76                5.offset = 41,19
77          }
78          MENU_LINKS.1.CUR < .MENU_LINKS.1.RO
79          MENU_LINKS.2 < .MENU_LINKS.1
80          MENU_LINKS.2 {
81                NO.backColor = #B0C0D0
82                NO.10.fontColor = #335C85
83                NO.20.file.backColor = #335C85
84                RO < .NO
85                RO.backColor = #FFFFFF
86                RO.5 >
```

```
87          }
```

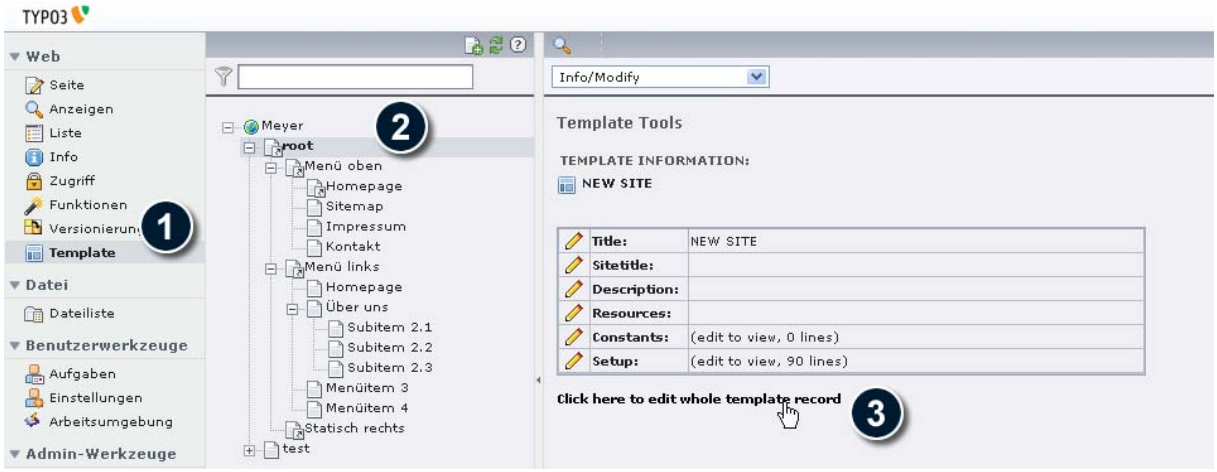
5.7 CONTENT: Inhalte ausgeben [mit css_styled_content]

Nachdem wir unsere Menüs erzeugt haben, können wir uns nun mit der Thematik "Inhalte ausgeben" beschäftigen. Im Kapitel 4.9 wurde bereits das Objekt CONTENT vorgestellt.

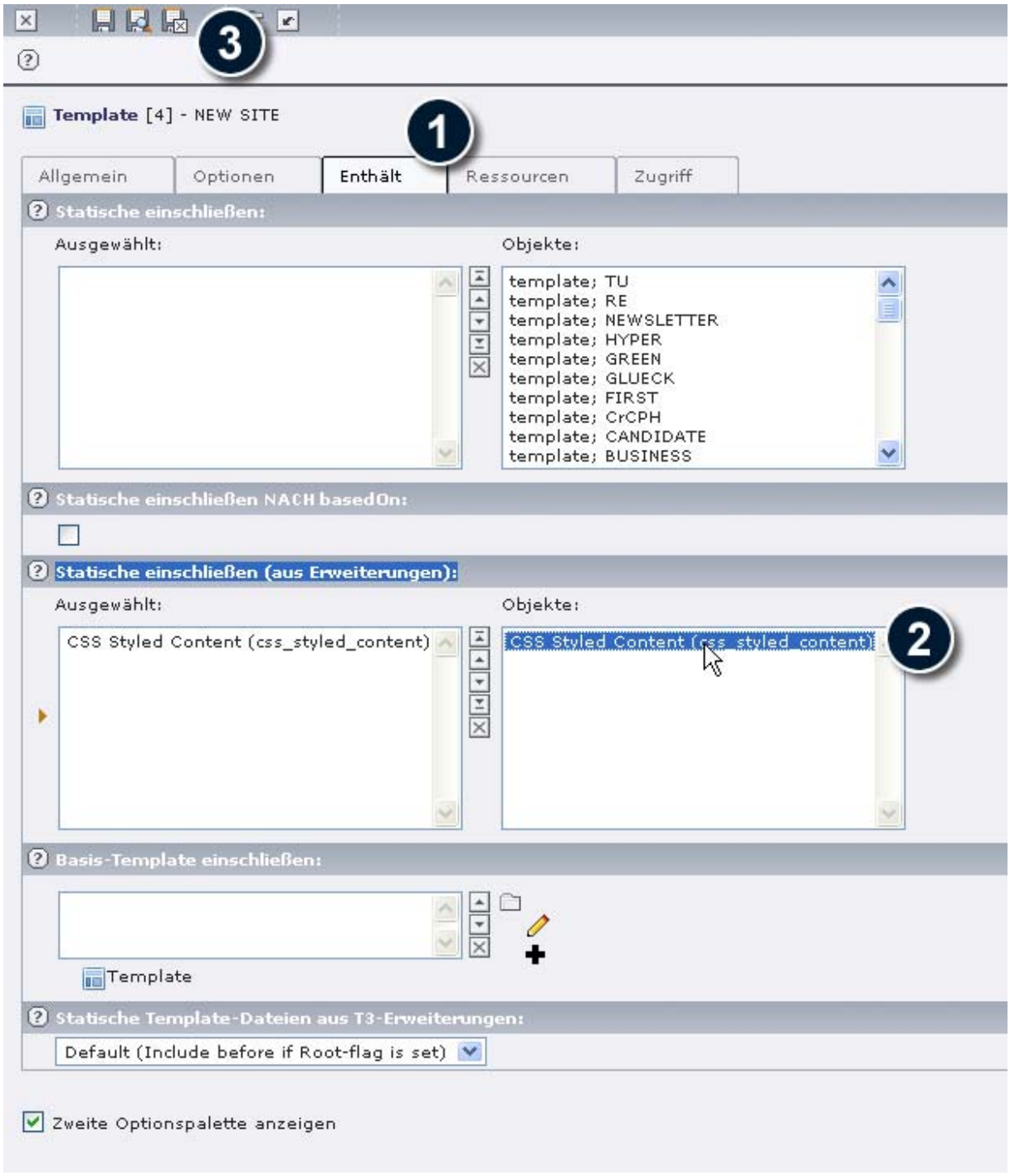
5.7.1 Vorbereitung: Statisches Template inkludieren

In Kapitel 4.9 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 so genanntes "statisches Template" mit dem Namen "css_styled_content". Hierzu klicken wir auf das Modul „Template“ und auf den Textlink der Seite „root“. Ganz unten ist der Textlink "Click here to edit whole template record" zu sehen, über den der gesamte Datensatz unseres Root-Templates bearbeitet werden kann:



Es öffnet sich eine umfangreiche Template-Maske. Nähere Informationen zu dieser Maske erhalten Sie im Kapitel 3.3. Im Reiter „Enthält“ klicken Sie im Abschnitt "Statische einschließen (aus Erweiterungen):" im rechten Feld auf "CSS Styled Content (css_styled_content)", um diese statische Template zu Ihrem Root-Template hinzuzufügen, und speichern den Datensatz ab.

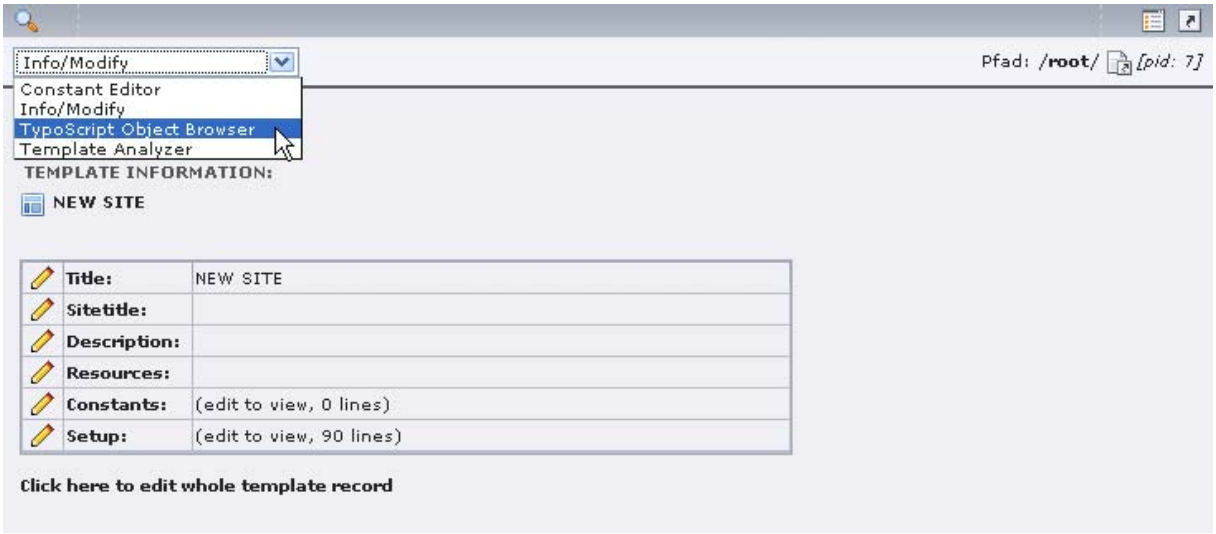


Neben CSS Styled Content steht aus älteren Zeiten noch das statische Template content(default) zur Verfügung. Von der Nutzung dieses Templates zur Ausgabe der Inhalte rate ich jedoch ab. Die Ausgabe ist sehr veraltet und enthält HTML-Code, der nicht mehr zeitgemäß ist.

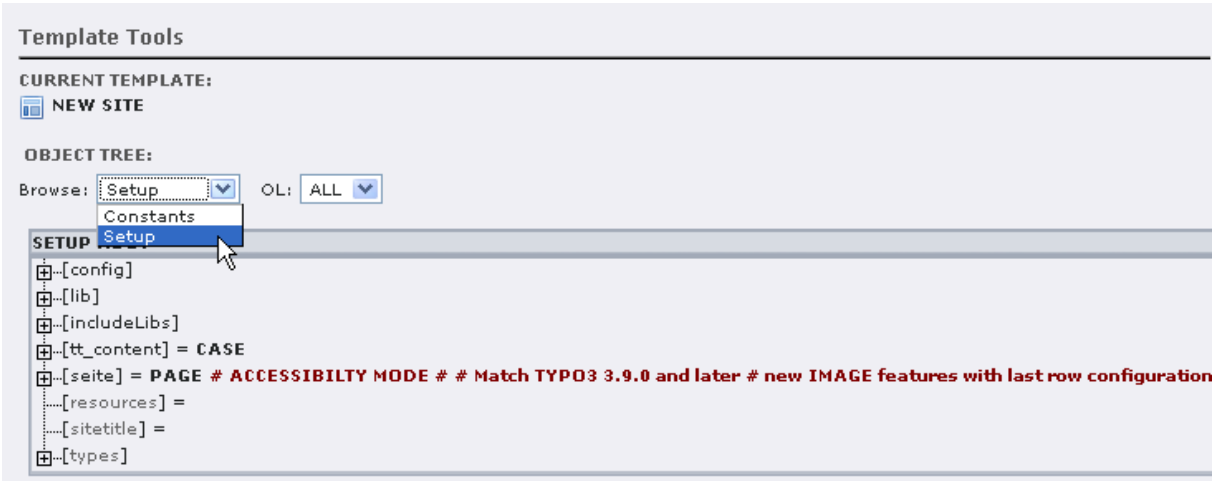
5.7.2 Analyse: css_styled_content unter die Lupe genommen

Wir nehmen nun das inkludierte statische Template unter die Lupe, um zu verstehen, was 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:



Es öffnet sich eine Baumdarstellung unseres Templates. Nähere Informationen im Kapitel 3.4. Beim erstmaligen Aufruf des Object Browsers ist der „***Object tree***“ auf Constants gesetzt. Dies bedeutet, dass alle Konstanten (Template Feld Constants) unseres Root Templates gezeigt werden. Wir möchten jedoch das Setup unter die Lupe nehmen. Prüfen Sie daher, ob die Auswahl bei ***Object Tree*** auf „***Browse: Setup***“ steht:

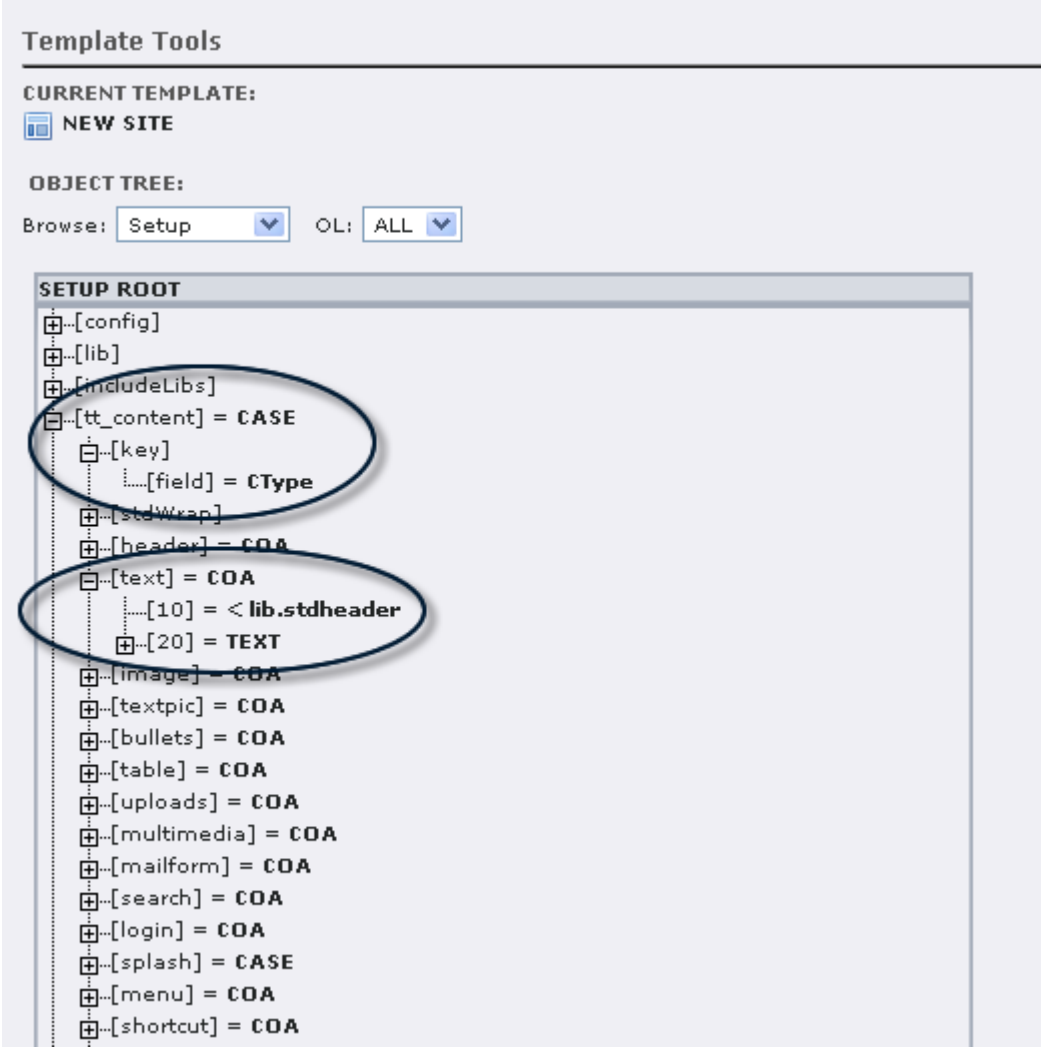


Gehen Sie, um zu sehen, was der Object Browser alles enthält, in den Abschnitt "***seite***" sowie die Unterseiten. Sie werden feststellen, dass hier Ihr TypoScript-Code vollständig abgebildet wird. Sie können auch im Object Browser editieren, indem Sie direkt auf eine Eigenschaft klicken. Das Resultat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

5.7.3 tt_content

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 4.5). 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 Referenz von "lib.stdheader" erfolgt. "lib.stdheader" enthält die Überschrift des Seiteninhaltes. Nähere Informationen hierzu folgen an späterer Stelle.

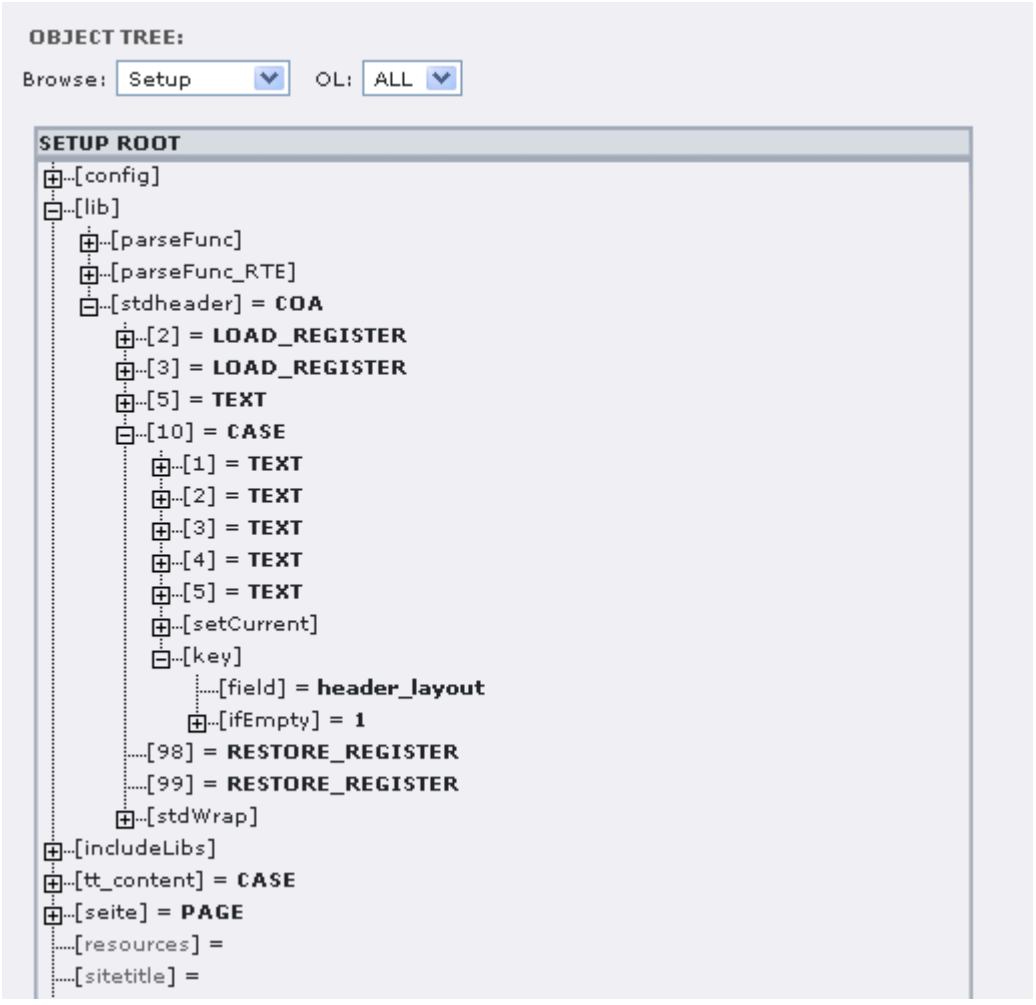
An der Position 20 wird eine Instanz des TEXT-Objektes erzeugt. Durch die Funktion ".field = bodytext" wird der in dem Datenbankfeld "bodytext" enthaltene Inhalt ausgegeben. Die Funktion "parseFunc" ermöglicht eine Manipulation und Überprüfung des im Datenbankfeld "bodytext" gespeicherten Inhaltes.

5.7.4 lib.stdheader

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

```
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 referenzieren 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 "css_styled_content“ 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
[...]
```

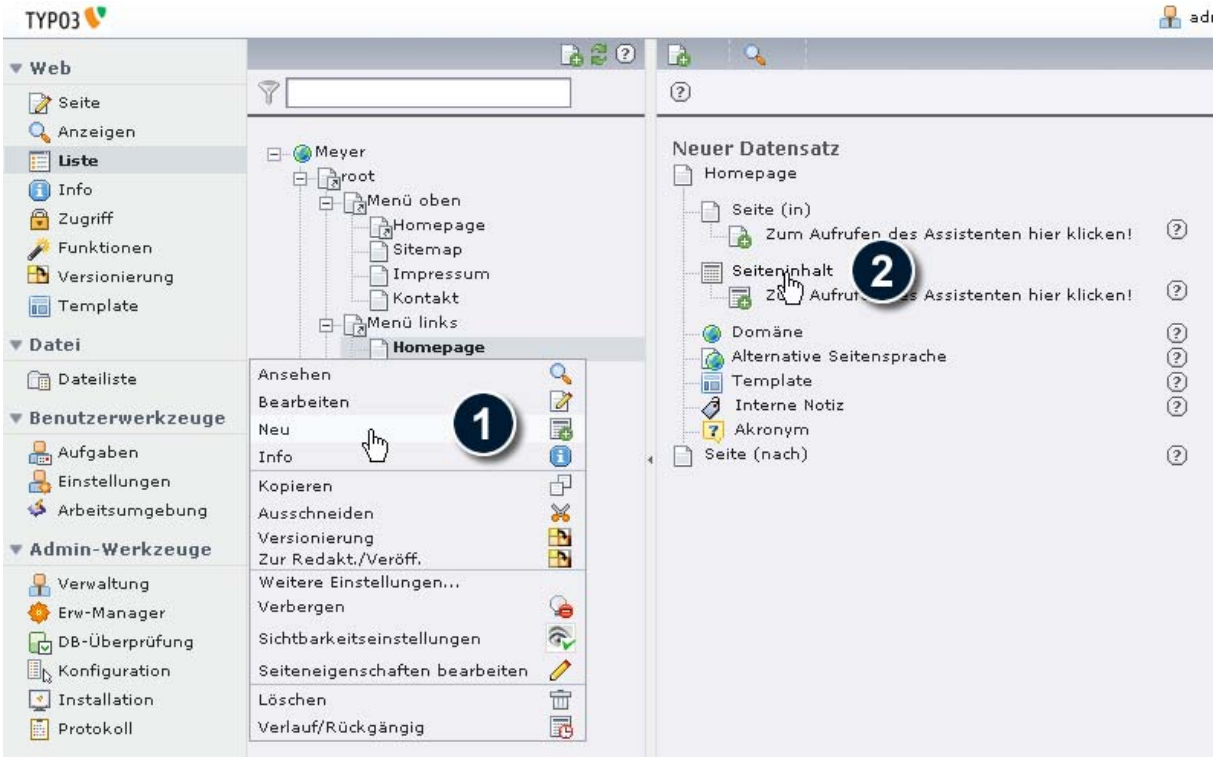
Die Überschrift besteht somit aus (mindestens) 2 Elementen: dem Element 5 und dem Element 10.

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).

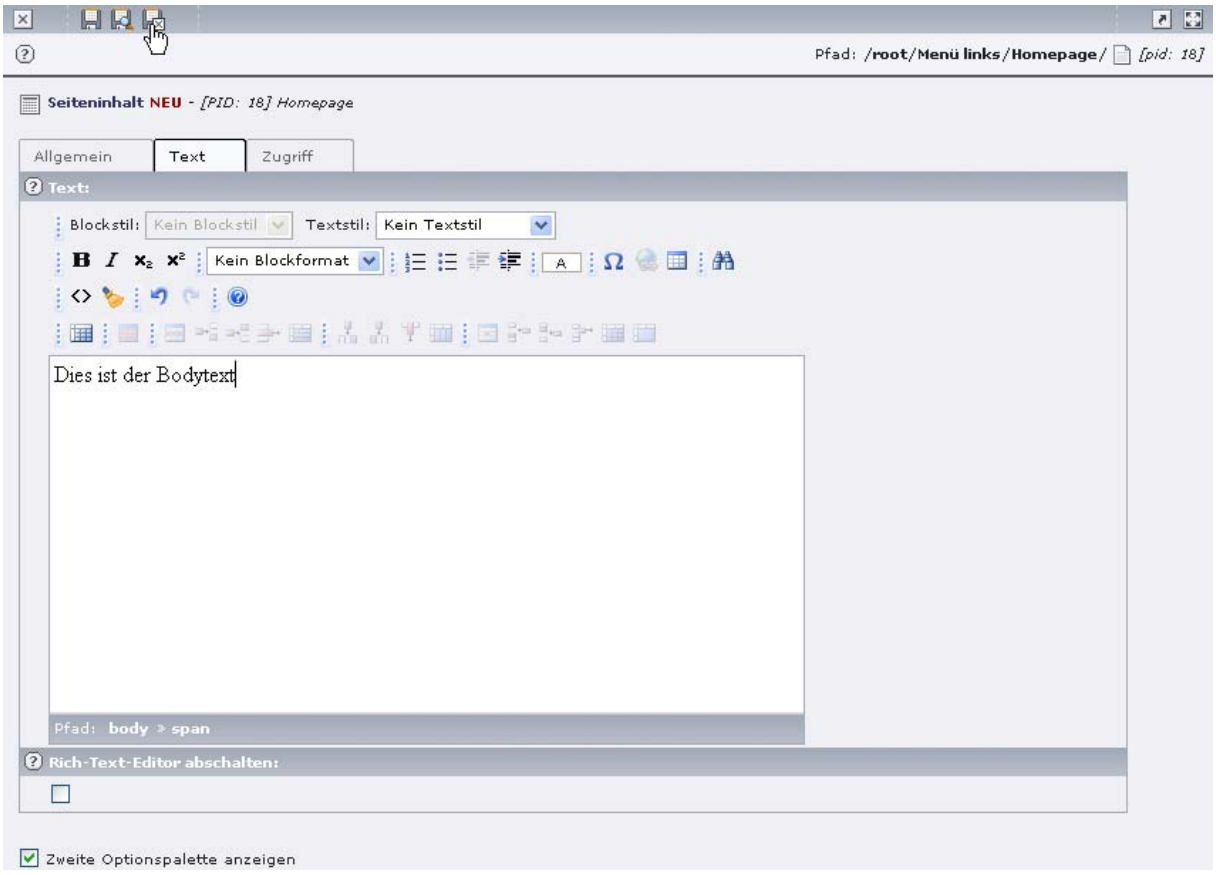
In lib.stdheader.10 wird eine CASE-Abfrage auf das Datenbankfeld header_layout vorgenommen (.key.field = header_layout). Die im Objekt Browser sichtbaren Abfragewerte und werden im Folgenden Instanzen. 1-5 sind Instanzen des TEXT-Objektes. Steht somit zu dem jeweiligen Datensatz im Datenbankfeld „header_layout“ z.B. eine „1“, wird der Abfragewert "1" verwendet und eine Instanz des TEXT-Objektes mit seinen jeweils angegebenen Eigenschaften gebildet. Es wird also eine Überschrift der ersten Ordnung mit einem font Tag <h1> erstellt:

5.7.5 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 Reiter „Allgemein“ im Feld „Überschrift“ den Text "Dies ist eine Überschrift" ein. Anschließen wechseln wir auf den Reiter „Text“ und geben im Feld „Text“ einen Inhalt an - "Dies ist der Bodytext". Anschließend speichern wir den Seiteninhalt, indem wir auf das Icon "Speichern und Schließen" klicken.



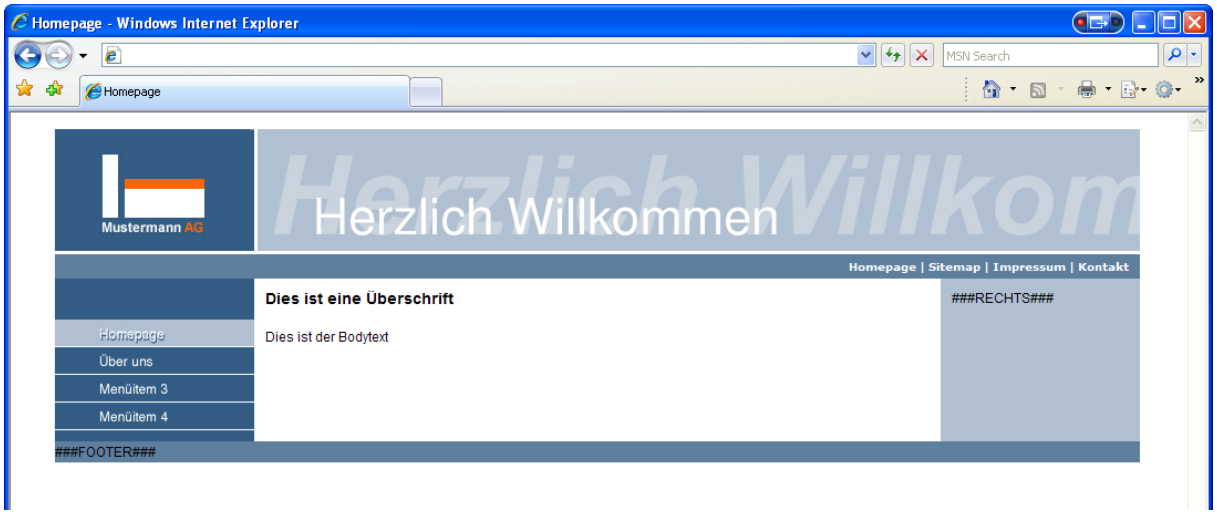
Ein Aufruf des Frontends wird uns jedoch zeigen, dass der soeben angelegte Inhalt noch nicht ausgegeben wird. TYPO3 „weiß“ ja noch nicht, wo es den Inhalt darstellen soll.

5.7.6 Objekt CONTENT verwenden

Wir erweitern nun unser Root-Template, damit wir den soeben angelegten Inhalt angezeigt bekommen. Der Inhalt soll an dem Marker "MITTE_CONTENT" dargestellt werden. Wechseln Sie in das Modul „Template“, klicken Sie auf die root-Seite und wechseln Sie in der Auswahlbox vom Object Browser wieder auf Info/Modify, um das Root-Template zu bearbeiten. Beachten Sie bitte, dass sich der folgende Abschnitt innerhalb der geschweiften Klammer von "seite.10.marks" befinden muss:

```
01 seite = PAGE
[...]
90 MITTE_CONTENT = CONTENT
91 MITTE_CONTENT {
92     table = tt_content
93     select.orderBy = sorting
94     select.where = colPos = 0
95 }
```

- In Zeile 90 wird auf dem Marker MITTE_CONTENT eine Instanz des Content-Objektes gebildet.
- In Zeile 92 wird angegeben, dass der Inhalt aus der Datenbank-Tabelle "tt_content" kommen soll.
- In Zeile 93 und 94 wird die intern erstellte SQL-Anweisung erweitert. Zum einen geben wir an, dass die Datensätze sortiert werden sollen (Zeile 93), zum anderen, dass nur Seiteninhalte angezeigt werden sollen, die in der Spalte "Normal/0" angelegt wurden.



5.7.7 Fehleranalyse: Es werden keine Inhalte dargestellt

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

- Wurde für das Template das statische Template "css_styled_content" inkludiert? (Kapitel 5.4.13)
- Überprüfen Sie, ob Sie wirklich die Seite im Frontend betrachten, auf der Sie einen Seiteninhalt angelegt haben, z.B. Homepage!
- Wenn der Marker noch angezeigt wird, haben Sie in TypoScript einen Fehler 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.

5.7.8 Darstellung anpassen: Überschrift

Die Darstellung der Überschrift können Sie per CSS formatieren, indem wir in der Datei style.css. die HTML Tags h1, h2 usw. anpassen. Es kann jedoch gewünscht sein, eine spezielle Schriftart als Überschrift zu nutzen, um diese beispielsweise an die Corporate Identity des Unternehmens anzupassen. Leider sind nicht alle Schriftarten aus allen Rechnern vorhanden, sodass eine Browser-Standardschrift genutzt werden würde.

Als Alternative kann die Überschrift als Grafik über den Gifbuilder generiert werden. Mit unserem bisherigen Wissen über lib.stdheader und den Gifbuilder sollte dies eigentlich keine Schwierigkeit sein.

Wir möchten, dass über das Auswahlfeld „Typ“ der Überschrift beim „Layout5“ eine Überschrift mit der Schrift „Arial Bold“ und einem Schlagschatten erstellt wird.

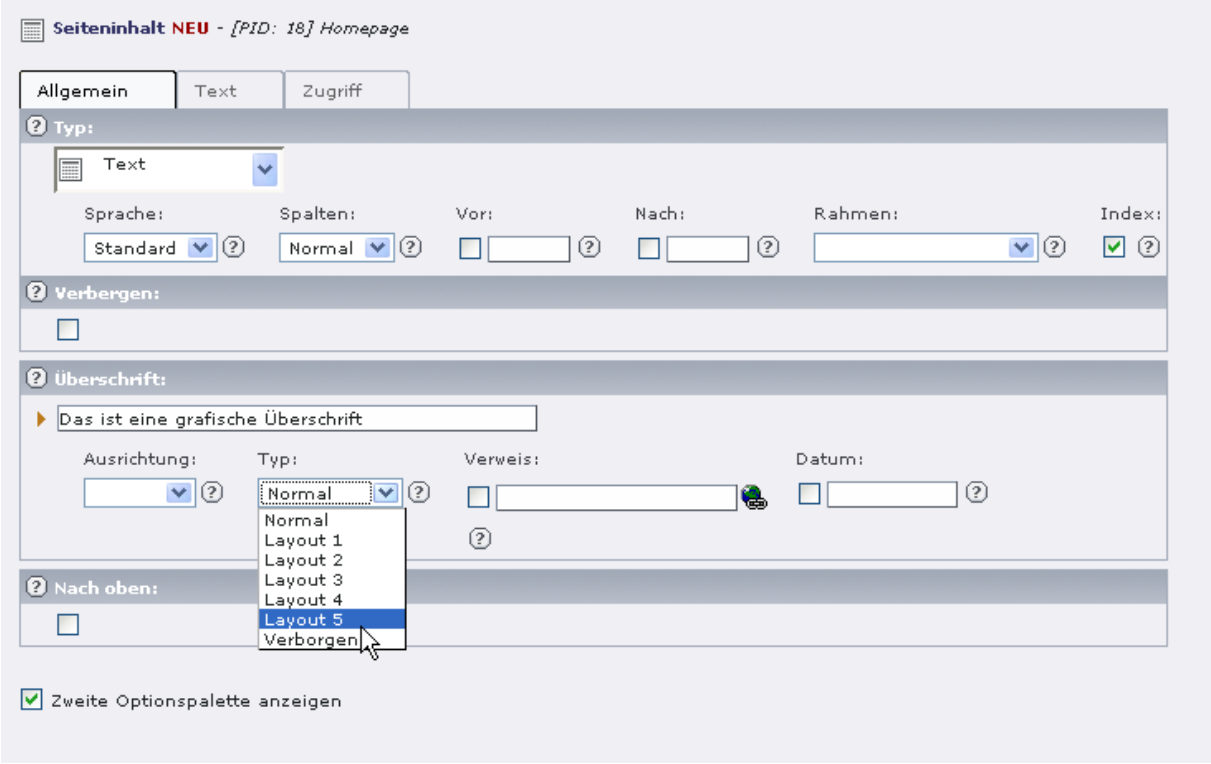
Wir erweitern bzw. überschreiben also lib.stdheader.10.5 mit einer IMAGE Instanz, deren Inhalt wir über den Gifbuilder erzeugen:

```
 01 seite = PAGE
    [...]
 98 # Eine grafische Überschrift (layout 5)
 99 lib.stdheader.10.5 = IMAGE
100 lib.stdheader.10.5{
101   file = GIFBUILDER
102   file {
103         XY = [10.w]+10, 22
104         backColor = #FFFFFF
105
106         10 = TEXT
107         10.text.field = header
108         10.fontFile = fileadmin/fonts/arialbd.ttf
109         10.fontColor = #061467
110         10.fontSize = 15
111         10.niceText = 1
112         10.offset = 1, 18
```

```
113
114         5 < .10
115         5.fontColor = #A9B0BD
116         5.offset = 3,20
117    }
118    wrap = |<br>
119}
```

- In Zeile 99 überschreiben erzeugen wir anstelle der Standard TEXT Instanz eine IMAGE Instanz
- In Zeile 101 erzeugen wir eine Datei über den Guifbuilder
- In Zeile 103 und 104 geben wir die Bildgröße und die Hintergrundfarbe an
- In Zeile 106 – 116 legen wir zwei Ebenen an, die Text- und die Schatten Ebene, und formatieren diese.
- In Zeile 118 fügen wir noch einen Zeilenumbruch ein.

Damit wir unsere Änderungen im Frontend prüfen können, legen wir auf der Seite Hompage einen Seiteninhalt an und wählen bei der Überschrift den Typ „Layout 5“ aus:



Im Frontend wird die grafische Überschrift dargestellt:

5.7.9 Darstellung anpassen: Bodytext

Die Darstellung des Inhalts wird über die hochgeladenen CSS Datei style.css gesteuert. Formatierungen sollten nur über CSS durchgeführt werden. Ein setzten von HTML Tags ist in der Regel nicht nötigt. Doch wo finden wir eine Übersicht der verwendeten CSS-Klassen, um diese in unserer CSS-Datei zu nutzen bzw. zu überschreiben?

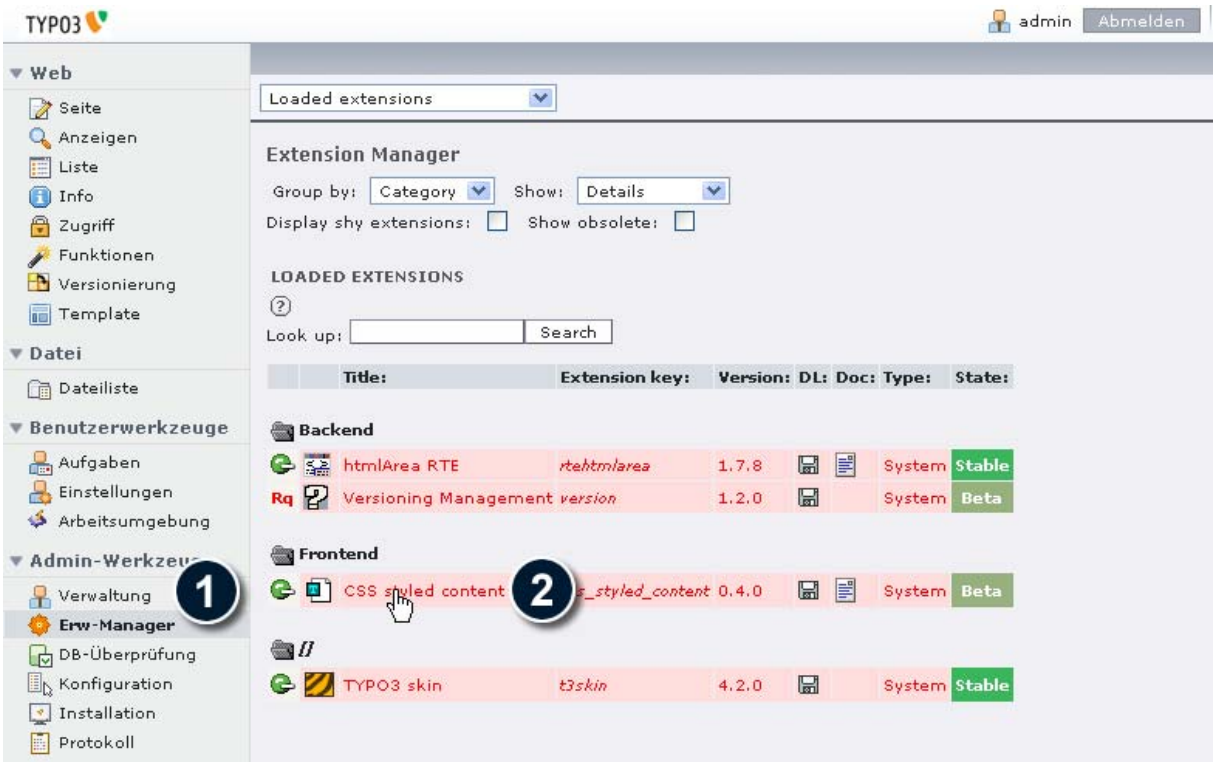
Es sind drei Wege möglich:

- Den Quelltext betrachten und die Klassen, die formatiert werden soll, suchen.
- Alle möglichen Bezeichnungen aus der Vorlage, die die System Extension CSS Styled content mitbringt, bearbeiten bzw. anpassen.
- Den Bezeichner der Klasse anpassen

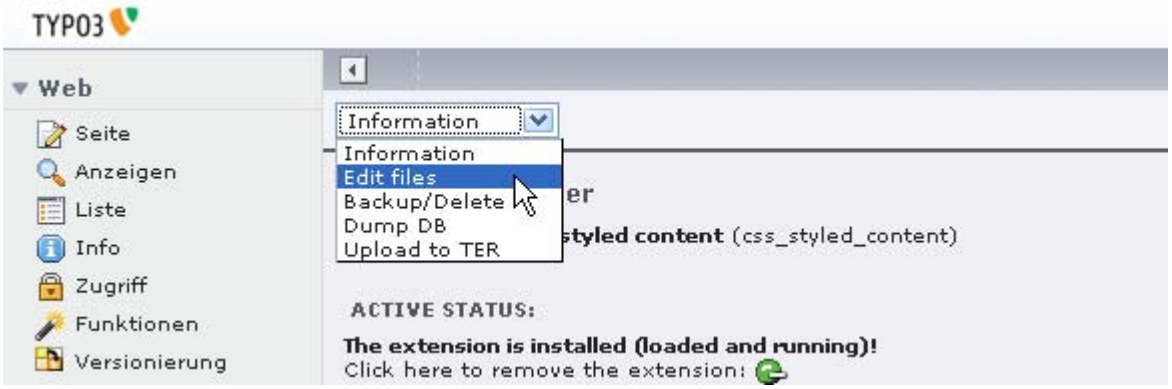
Ich möchte nicht auf alle drei Wege eingehen, jedoch zeigen, wo Sie eine Beispiel-Datei einsehen können, die als Vorlage zur Formatierung der CSS-Klassen von CSS Styled Content genutzt werden kann.

Gehen Sie in das Modul „Erweiterungs-Manager“ und klicken Sie auf den Textlink der Extension „CSS styled content“.

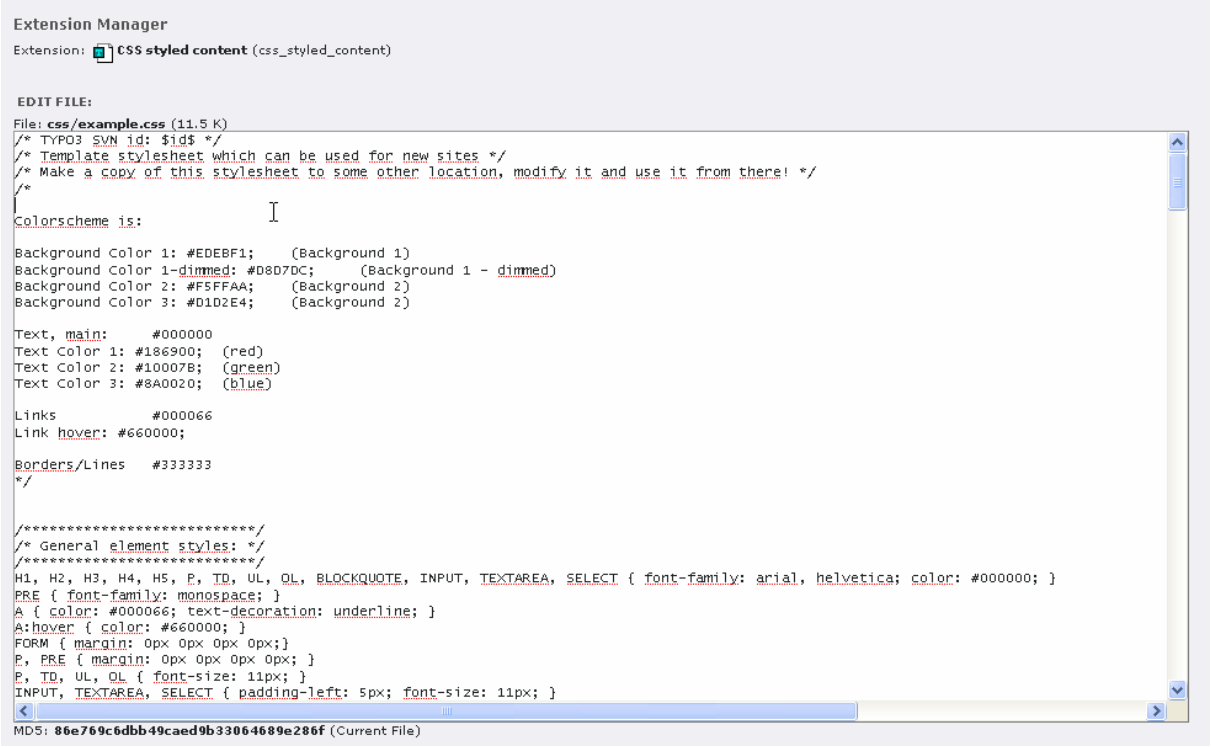
Achtung, klicken Sie nicht auf das grüne Icon – Minuszeichen vor der Extension, da Sie diese sonst deinstallieren.



Wählen Sie „Edit Files“ aus der Auswahlbox im oberen Teil der Maske aus:



Sie erhalten eine Übersicht aller Dateien, die die Systemerweiterung „CSS styled content“ beinhaltet. Interessant ist die Datei example.css, die Sie über den Textlink „Edit file“ betrachten und deren Inhalt auch kopieren können:



Sie können die Style Sheets in die Datei style.css übernehmen und so alle Inhalte nach Ihren Wünschen anpassen.

5.7.10 Rechte Spalte: Inhalte darstellen

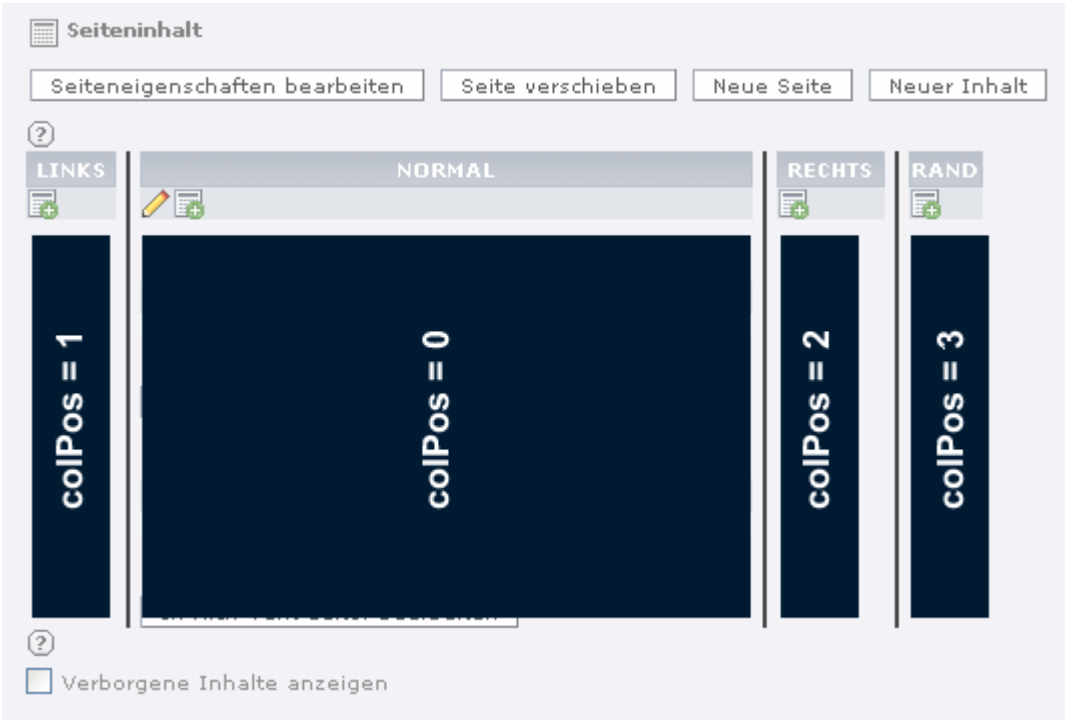
Der Marker "###RECHTS###" wird ähnlich dargestellt wie der Marker ###CONTENT###. Einziger Unterschied ist die Angabe der Eigenschaft select.where des CONTENT-Objektes.

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

Fügen wir zum Testen auf unserer Seite „Homepage“ in der rechten Spalte ein Inhaltselement vom Typ „Text“ ein.

5.7.11 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:



6. Module und eigene Erweiterungen

6.1 Einführung

TYPO3 ist vollständig modular aufgebaut. Seit der Version 3.5.x wurde TYPO3 um komfortable Modul-Funktionalitäten erweitert, die dem CMS einen großen Schub nach vorne gegeben haben. Eigene Modulentwicklungen lassen sich nun komfortabel durch einen Klick (oder durchaus auch mal mehrere) installieren.

In dem so genannten "Extension Repository - TER" können Module zentral abgelegt und der Öffentlichkeit zur eigenen Verwendung zur Verfügung gestellt werden. Dieses Prinzip 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. Seit der TYPO3 Version 4.0 werden Extensions von einem Security-Team auf Sicherheit geprüft. Eine Garantie, dass die Module fehlerfrei und sicher sind, gibt es jedoch nicht.

Ebenso kann es seit der Version 3.6.x zu Kompatibilitätsproblemen kommen: Module setzen eine ganz bestimmte TYPO3-, PHP- oder MYSQL-Version voraus. Steht diese bestimmte Version nicht zur Verfügung, so kann es bei dem einen oder anderen Modul (insbesondere bei Modulen im Alpha-, Beta- oder Obsolete-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.

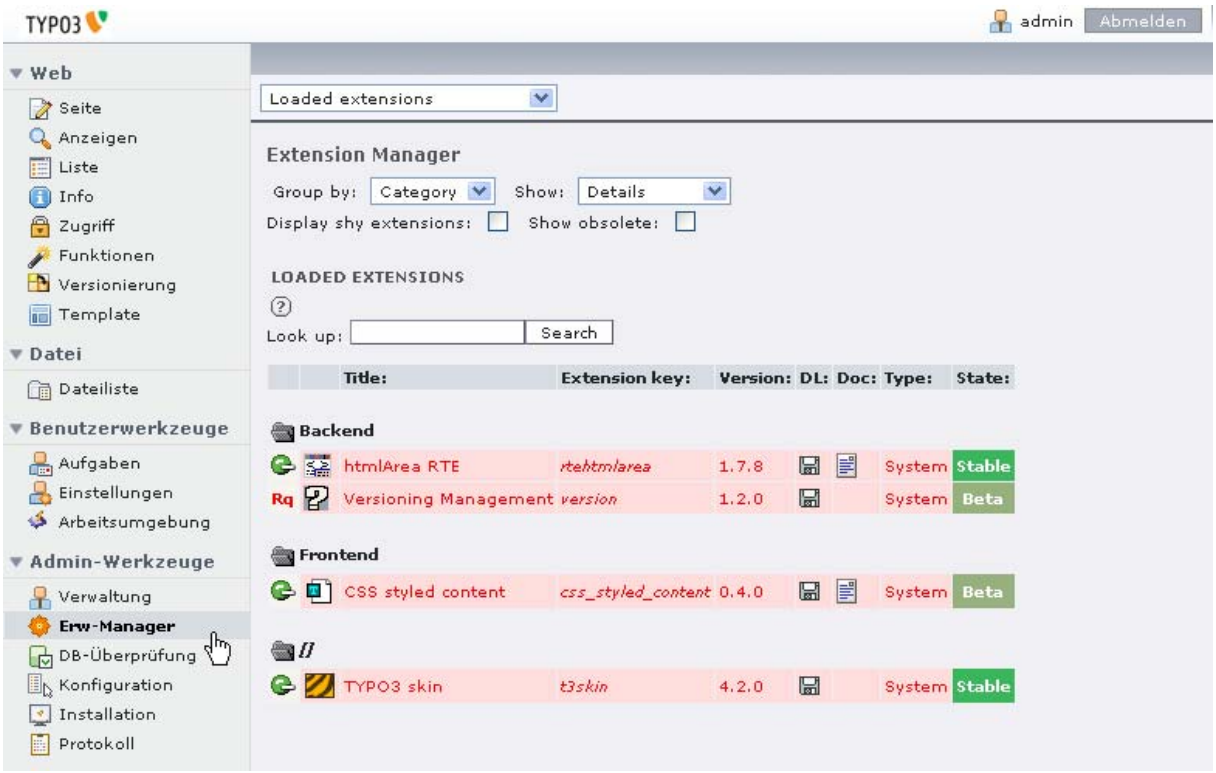
Diese Einführung soll nicht davor abschrecken, Module zu verwenden. Jedoch sollten Sie Module nicht in einer Live-Umgebung testen, da dieses unangenehme Folgen haben kann. Testen Sie Module daher möglichst in einer separaten Testumgebung auf Fehler.

6.2 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:

6.2.1 Shy und Obsolete Extensions

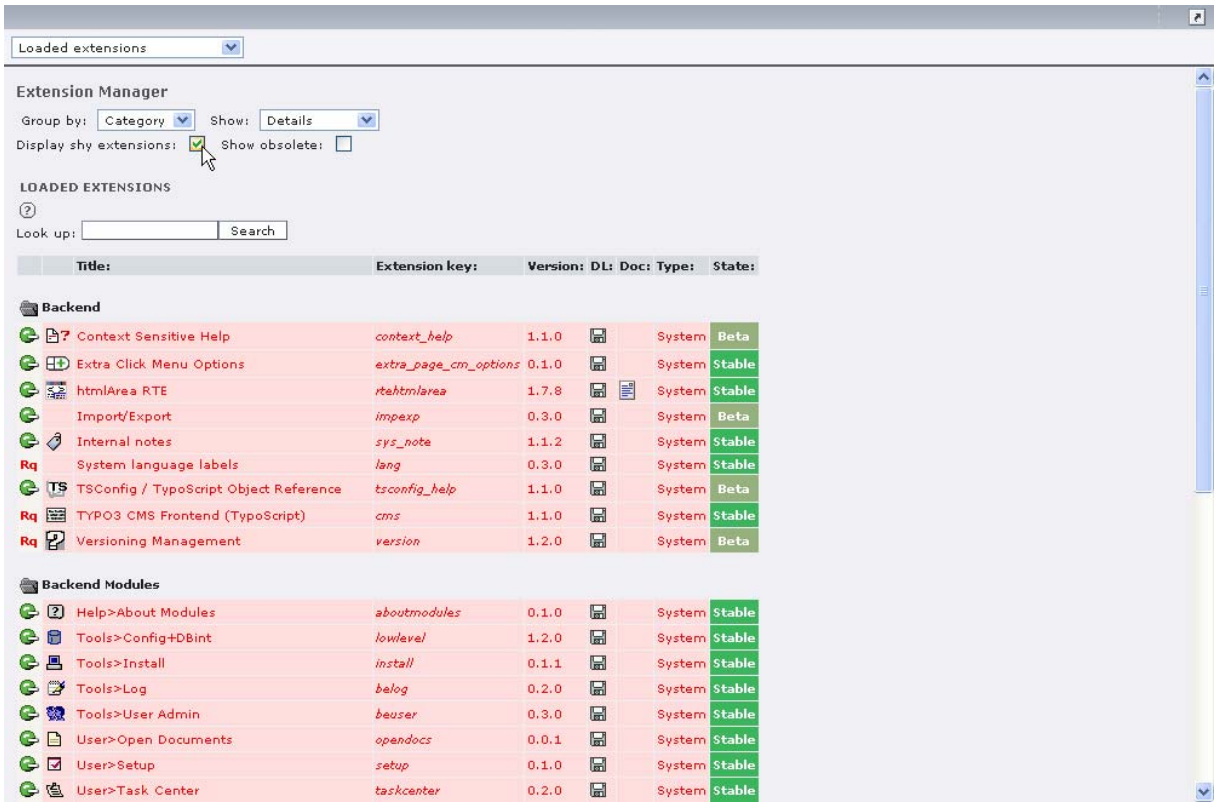
Wenn wir den Erweiterungs Manager aufrufen, sehen wir zunächst eine Übersicht der "geladenen" Module.



Wie im Screenshot zu sehen, scheinen jedoch nur wenige Module installiert zu sein. Der Schein trügt allerdings. Wie bereits in der Einleitung zu diesem Kapitel erwähnt, ist TYPO3 selbst 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. Dass jedoch z.B. Menüpunkte wie Dateiliste, Log oder auch Template 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.

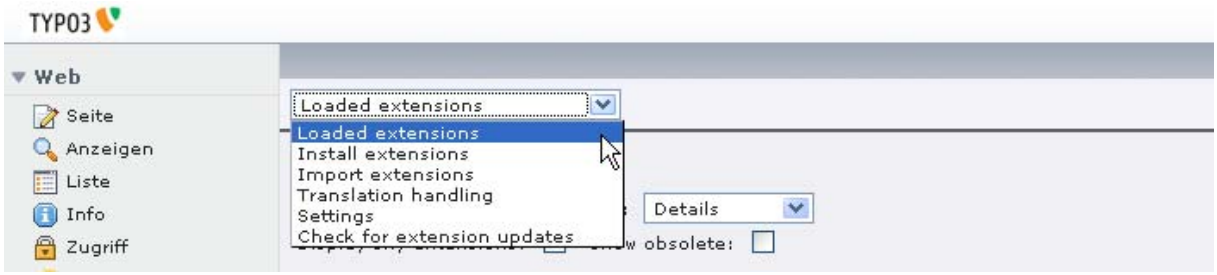
Damit wir diese "versteckten" Extensions angezeigt bekommen, können wir uns diese durch Setzen des Häkchens im Feld "Display shy extensions" anzeigen lassen:



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

6.2.2 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.
- **Install extensions:** 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, /typo3/sysext oder von /typo3conf/ext befinden.

- **Import extensions**: 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.
- **Translation Handling**: Über das Menü werden Sprachpakete geladen, im Kapitel 2.6.2 haben Sie hierüber bereits die deutsche Backendsprache gealden.
- **Settings:** In den Einstellungen können Sie einen Mirror angeben, den Sie für den Download von Erweiterungen aus dem TER nutzen möchten. Sie können Einstellungen durchführen, die Sie für das Veröffentlichen von Erweiterungen benötigen. Eine wichtige Einstellung ist die De-/Aktivierung der Sicherheitsüberprüfung bei Erweiterungen.
- **Check for extension Updates:** Seit der TYPO3 Version 4.2 steht diese nützliche Funktion zur Verfügung, mit der alle genutzten Erweiterungen auf Updates überprüft werden können.

6.2.3 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 einen 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.

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 verwendet 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.

In der sechsten Spalte steht die Erweiterung als Download link, über das Disketten-Symbol zur Verfügung. So können Sie z.B. eine Sicherung herunterladen.

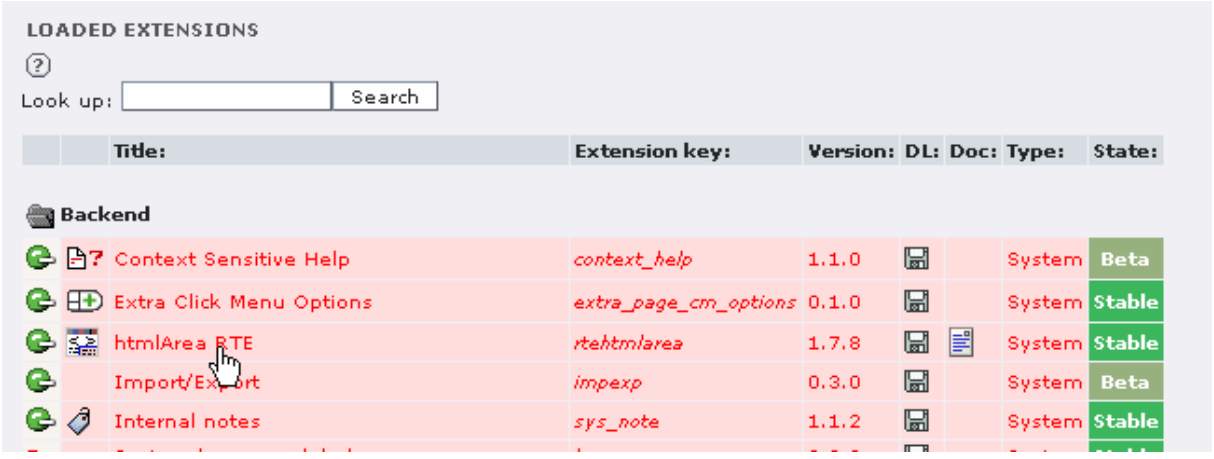
Die siebte Spalte zeigt an, ob zu diesem Modul eine Dokumentation zur Verfügung steht.

In der achten Spalte "Type" können Sie erkennen, wo das Modul installiert wurde. Dies ist insbesondere für spätere Updates von der TYPO3-Installation interessant. Es gibt 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 6.2.

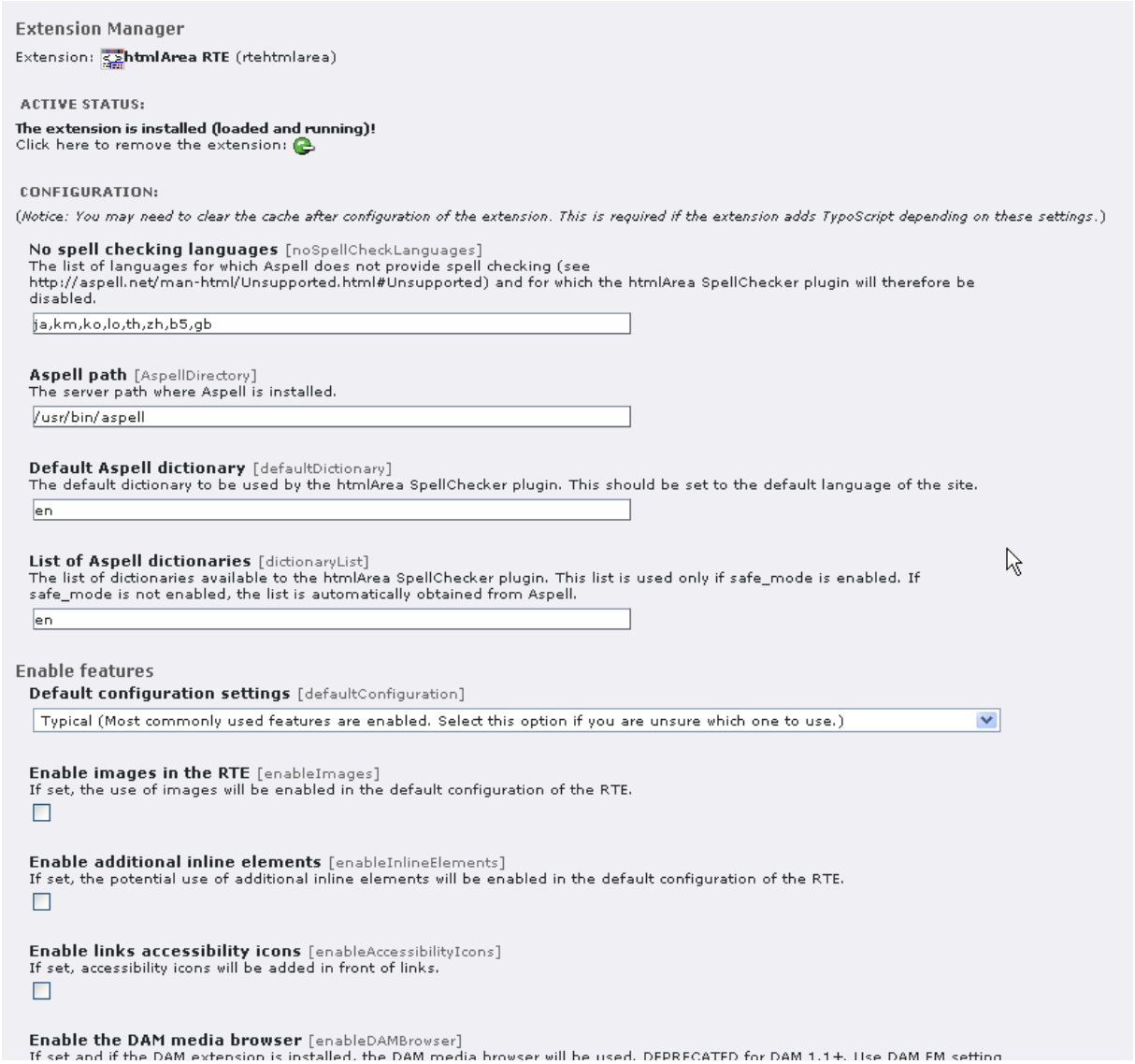
In der neunten Spalte wird der Status des Moduls angegeben. Mögliche Werte sind Test, Experimental, Obsolte, Alpha, Beta und Stable. Der Status wird jedoch nicht von einem Gremium vergeben, sondern direkt vom Entwickler des Moduls selbst. Module mit dem Status "Stable" können also durchaus noch Fehler enthalten, da sie z.B. nur für eine bestimmte TYPO3-Umgebung getestet wurden.

6.2.4 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.

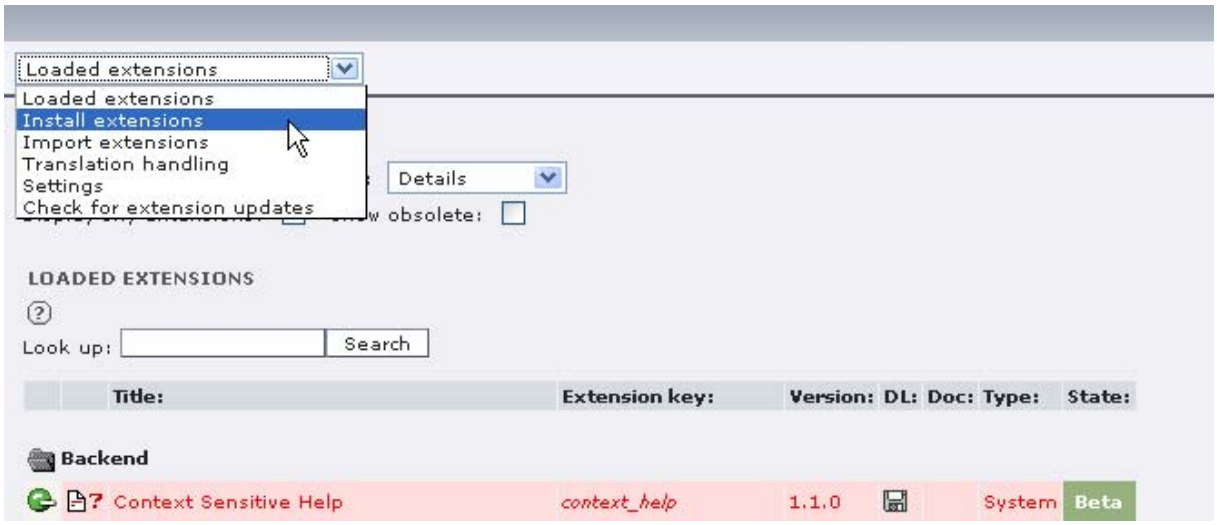


Aus der Auswahlbox links oben können noch weitere Bereich ausgewählt werden, wie z.B. der im Kapitel 5.4.21 bereits gezeigte Menüpunkt "***Edit Files***".

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.

6.2.5 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 "Install extensions" 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 endgültig zu installieren.

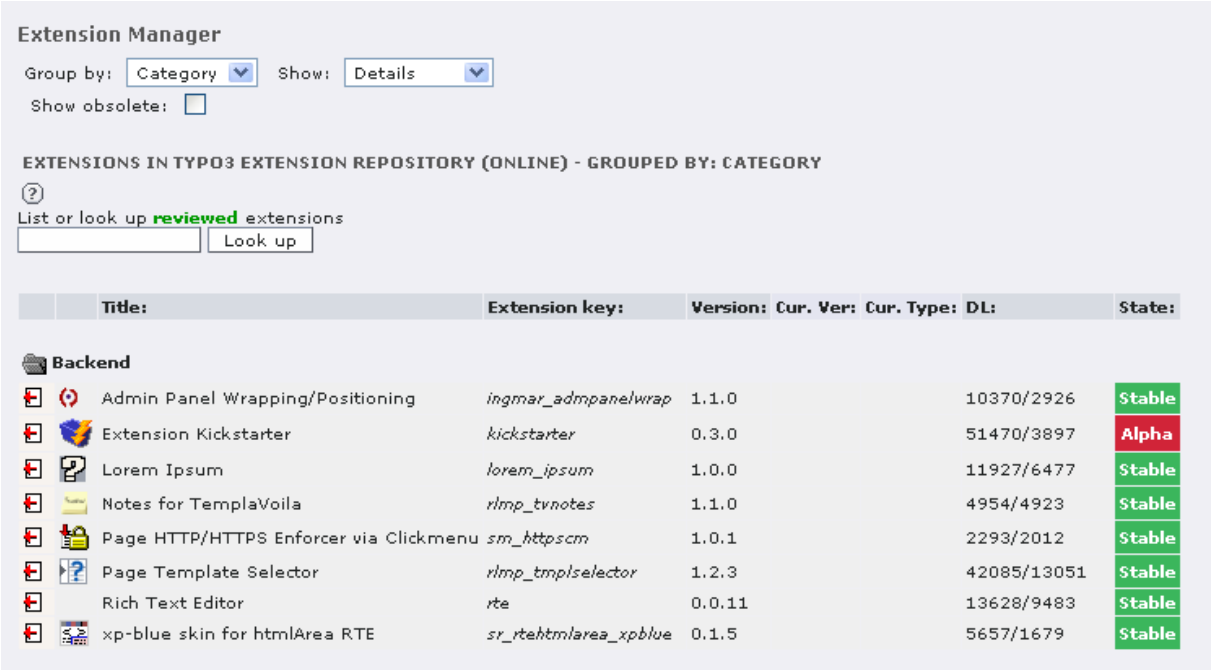
6.2.6 Module von der Extension Repository herunterladen

Steht eine Erweiterung nicht zur Verfügung, so können Sie diese aus dem Extension Repository herunterladen. Das Extension Repository ist ein zentraler Server, der als Sammelpunkt aller veröffentlichten Module dient.

Um eine Verbindung zu diesem zentralen Server aufzubauen, wählen Sie aus der Auswahlbox den Eintrag "Import extensions“ aus.

Um eine aktuelle Liste aller im TER verfügbaren Extensions zu erhalten, klicken wir auf die Schaltfläche „Retrieve/Update“. Der Ladevorgang kann unter Umständen einige Sekunden benötigen, anschließend erhalten wir eine Meldung, dass die Liste aktualisiert wurde.

Um die Liste einzusehen, klicken wir auf die Schaltfläche „Look up“.



Sollten Sie nicht alle Erweiterungen aus dem TER angezeigt bekommen, kann dies an der Sicherheitsüberprüfung innerhalb von TYPO3 liegen. Möchten Sie, dass auch Erweiterungen angezeigt werden, die nicht vom Security Team geprüft worden sind, so gehen Sie in den Menüpunkt **„Settings“** und aktivieren Sie die Option **„Enable extensions without Review“**.

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".

6.2.7 Generelles zur Updatefähigkeit

Sie möchten, dass Ihre Erweiterungen nach einem TYPO3 Update weiter zur Verfügung stehen? Dann haben Sie gedanklich schon eine gute Voraussetzung dafür geschaffen, dass Sie später, sofern Sie denn TYPO3-Updates tätigen möchten, nicht vor unschönen Problemen stehen, die oftmals erst nach einem getätigten Update auftreten.

Wie bereits in Kapitel 6.1 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 gleichzeitig liegen. Module, die im Ordner /typo3conf/ext liegen, sind updateresistent.

Folgender Hintergrund:

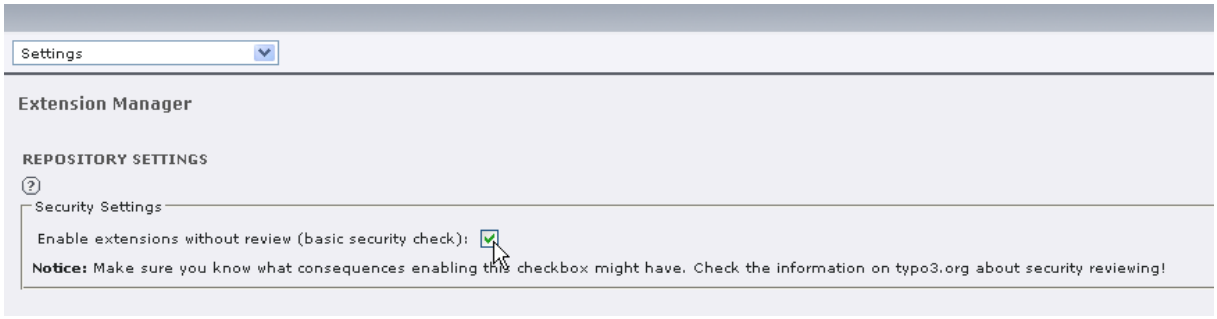
Bei einer "leeren" TYPO3-Installation werden bereits viele System-Module mitgeliefert. Diese sind im Verzeichnis typo3/sysext installiert. Dies deutet darauf hin, dass diese Module Bestandteil des Original-Quelltextes sind, der bei einem Update entweder überschrieben oder ausgetauscht wird. Das Verzeichnis für die globale Installation von Erweiterungen liegt ebenfalls innerhalb des Verzeichnisses TYPO3 (typo3/ext/). Installieren Sie also Extensions global, würde dieser Ordner bei einem TYPO3 Update ausgetauscht werden und das Modul stünde nicht mehr zur Verfügung. Wenn ein Modul updatefähig gehalten werden soll, dann muss sich das Modul im "lokalen Ordner" /typo3conf/ext befinden.

Die globale Installation von Erweiterungen ist als Standrad seit TYPO3 Version 4 deaktiviert. Sie können dies im TYPO3 Install Tool aktivieren, sollten jedoch immer die lokale Installation ausführen.

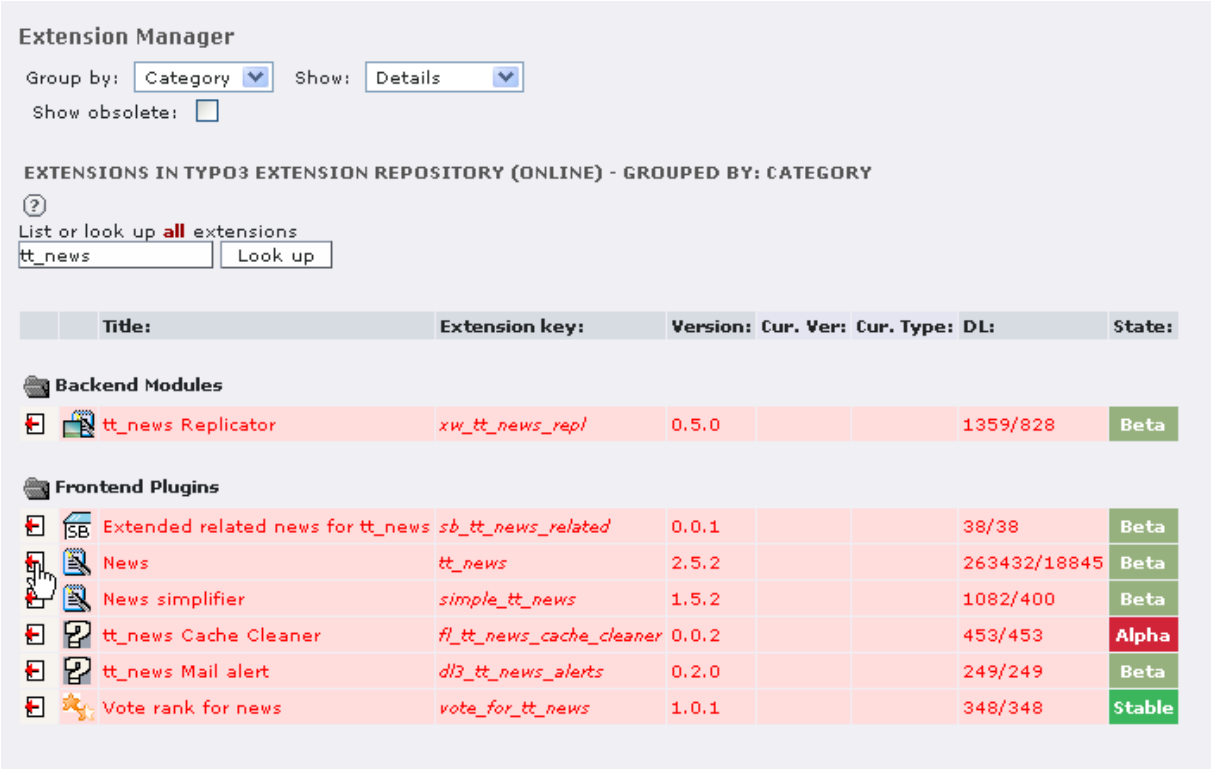
6.2.8 Frontend-Plugins integrieren und anpassen

6.2.9 Das News-Plugin installieren

Um das Newsmodul zu installieren, wählen wir im Erweiterungsmanager aus der Auswahlbox den Eintrag „***Settings***“ aus und deaktivieren die Sicherheitsüberprüfung. Dies ist sinnvoll, um sicherzustellen, dass wir die neuste Version der Extension erhalten.

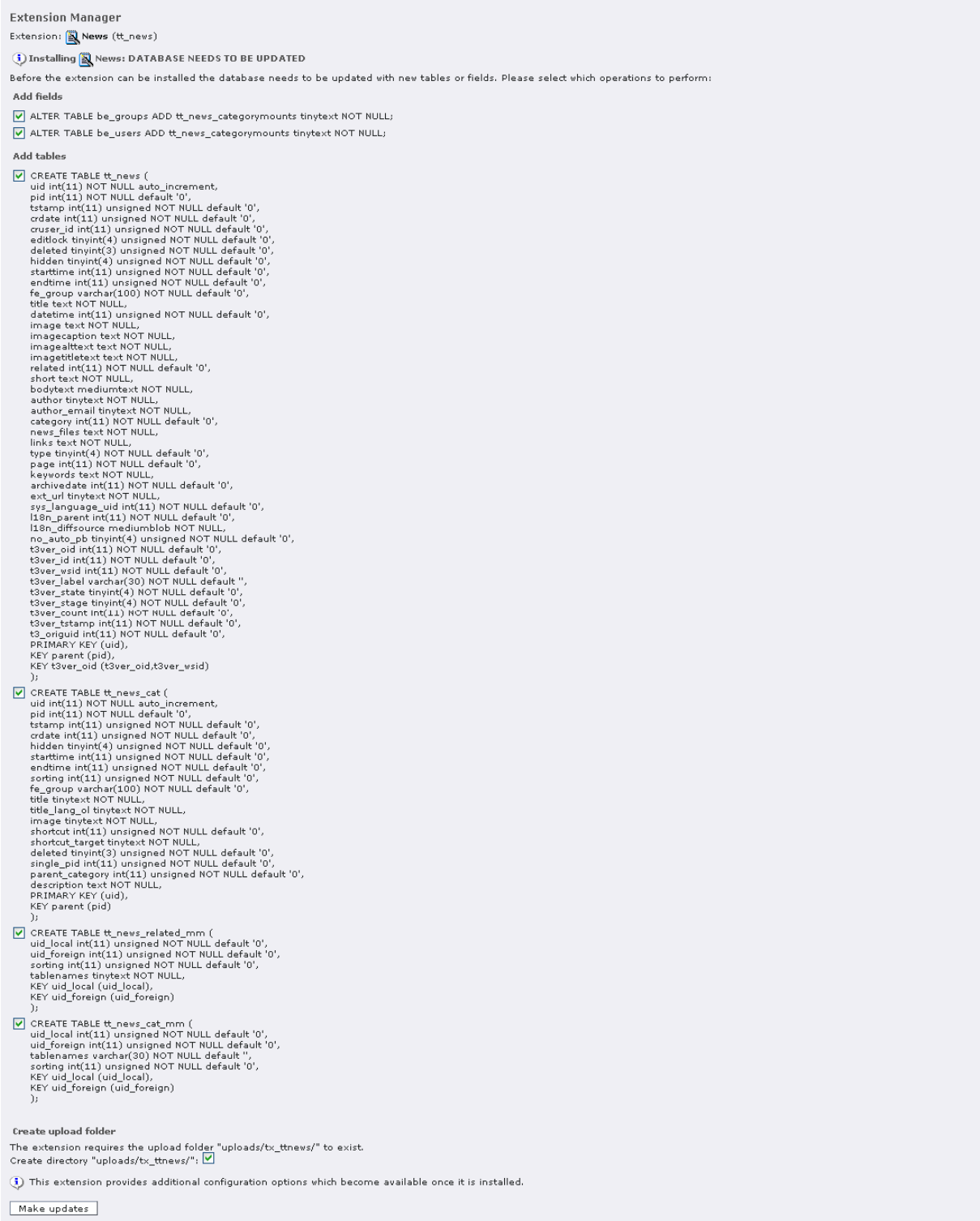


Anschließend wechseln wir in das Menü "***Import extensions***". Wir geben im Feld „***List or look up reviewed extensions***“ den Begriff “***tt_news***” ein und klicken auf die Schaltfläche “***Look up***”. Wir suchen aus der Liste die Extension mit dem Extensionkey **tt_news** heraus und laden die Extension auf unseren Server, indem wir auf das Icon mit dem roten Pfeil klicken.



Nach der Übertragung folgt eine Seite, auf der wir gefragt werden, ob wir die Erweiterung installieren möchten. Wir klicken auf das graue Plus-Symbol vor dem Text „***Install extension***“.

Es folgt eine Maske, 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:

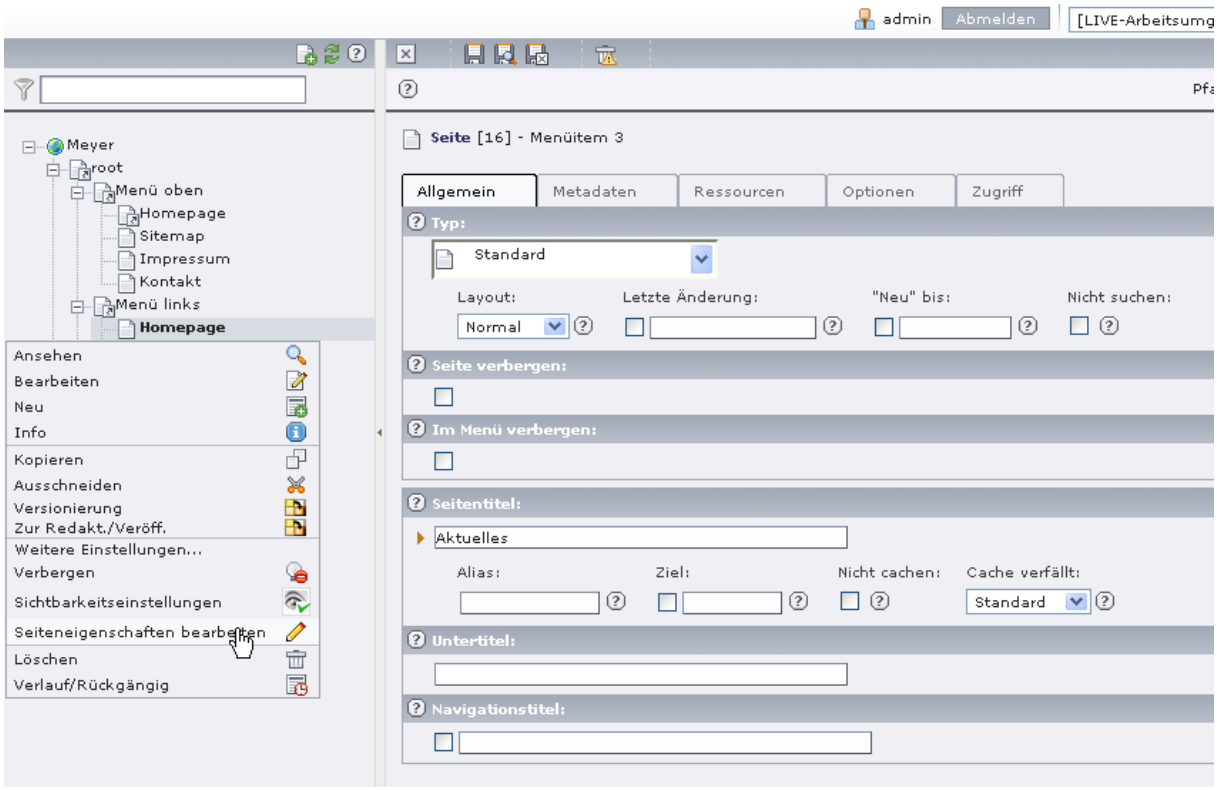


Das News-Modul wurde nun installiert – eine direkte Veränderung im Backend lässt sich auf den ersten Blick nicht erkennen.

6.2.10 Frontend-Plugin: Seiteninhalt/Container anlegen

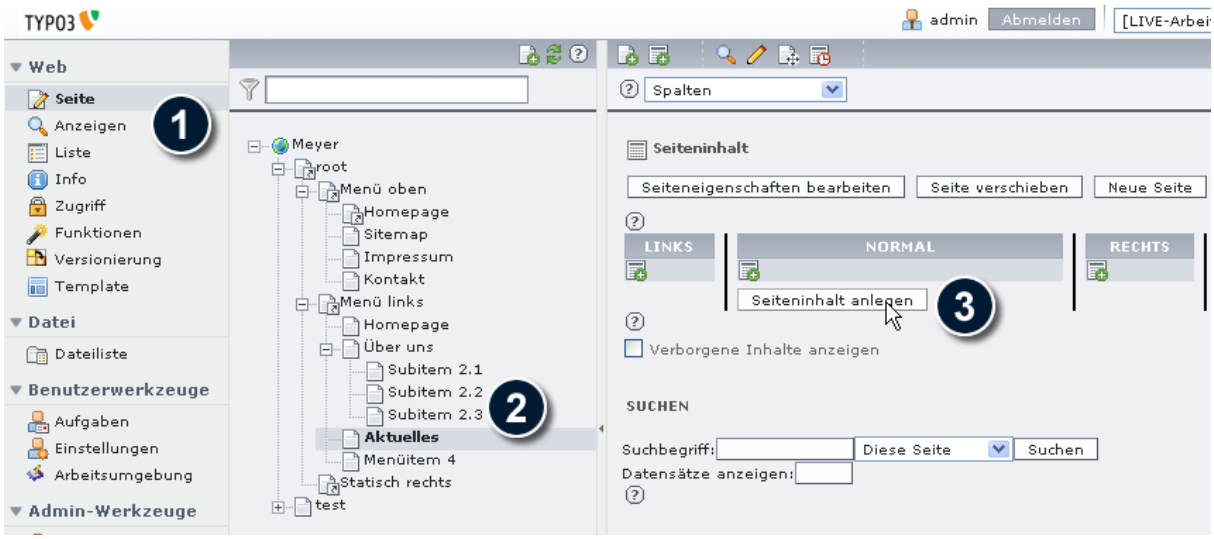
Wir können nun auf einer beliebigen Webseite unser News-Modul hinzufügen. Eine brauchbare Seite wie z.B. "Aktuelles" gibt es derzeit noch nicht – wir benennen darum die Seite "Menüitem 3" in "Aktuelles" um. Hierzu klicken wir im Seitenbaum auf das Icon, wählen aus dem Popup-Menü

den Eintrag "Bearbeite Seiten Header" aus und bearbeiten den Seitentitel in der sich öffnenden Maske auf der rechten Seite.

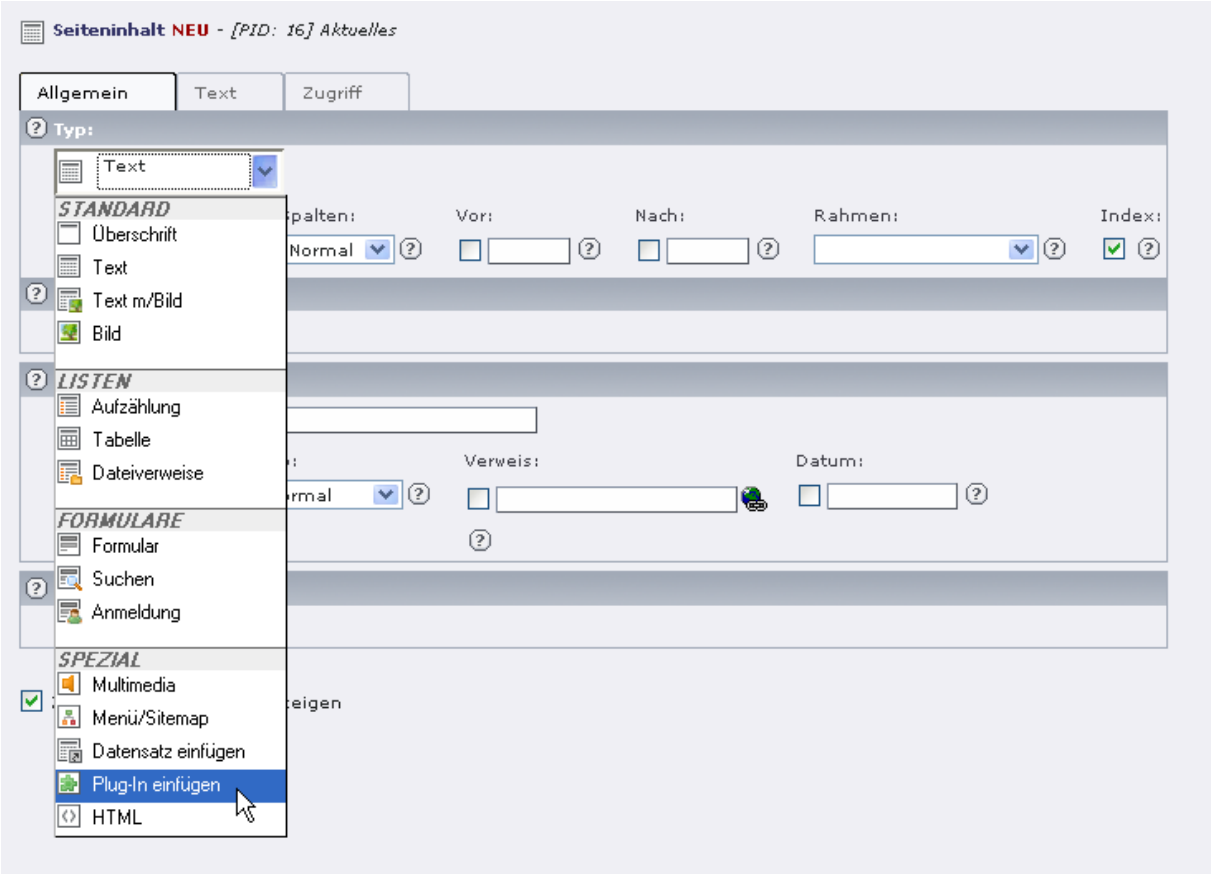


Nachdem wir die Seite mit neuem Seitentitel gespeichert haben, wird uns der neue Seitentitel ebenfalls im Seitenbaum angezeigt.

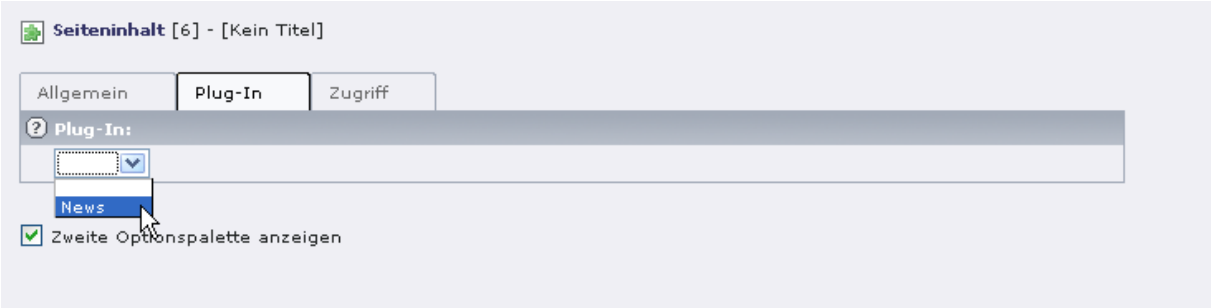
Auf dieser Seite "Aktuelles" können wir nun einen Seiteninhalt ablegen. Hierzu wählen wir im linken Menü den Eintrag "Seite" und im Seitenbaum unsere umbenannte Seite "***Aktuelles***" aus.



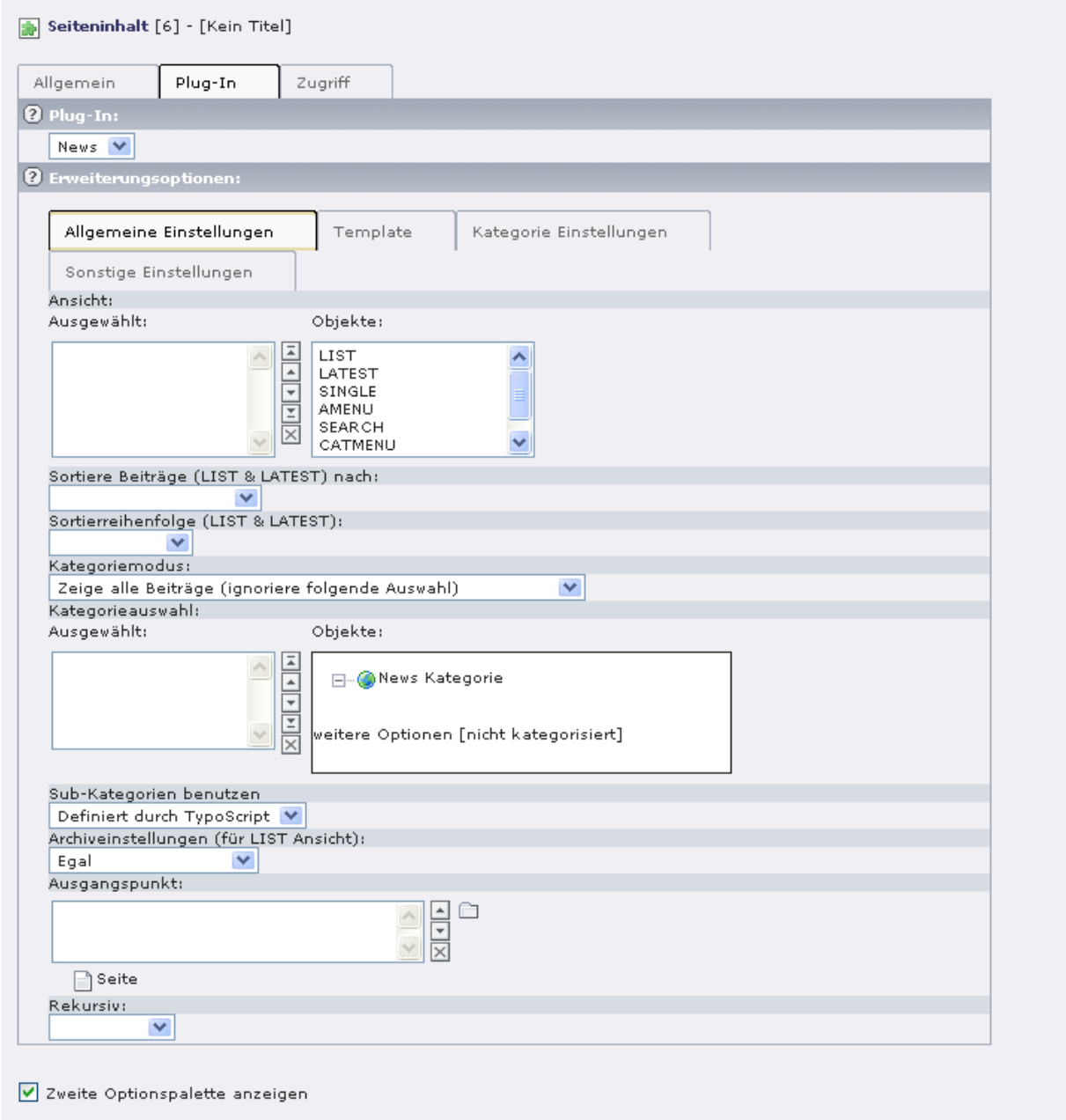
Durch Anklicken des Buttons "***Seiteninhalt anlegen***" können wir nun einen Seiteninhalt anlegen. Es öffnet sich der "Wizard" für neue Seiteninhalte: Sie können entweder einen ganz normalen Seiteninhalt wie z.B. "Normaler Text" auswählen oder aber im unteren Bereich "***Erweiterungen***" den Eintrag "***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, und bestätigen Sie die angezeigte Meldung mit „***OK***“.



Sie erhalten jetzt eine Maske für Plugins, in der ein neuer Reiter „Plug-In“ angezeigt wird. Sie können im Reiter „Allgemein“ eine Überschrift vergeben. Wechseln wir anschließend auf den Reiter „Plug-In“. Dort wählen wir das gewünschte Frontend-Plugin aus der Liste der installierten FE-Plugins aus. In unserem Fall sollte die Liste nur einen Eintrag "News" enthalten. Wählen Sie als Plug-In „News“ aus und bestätigen Sie angezeigte Meldung mit „OK“.



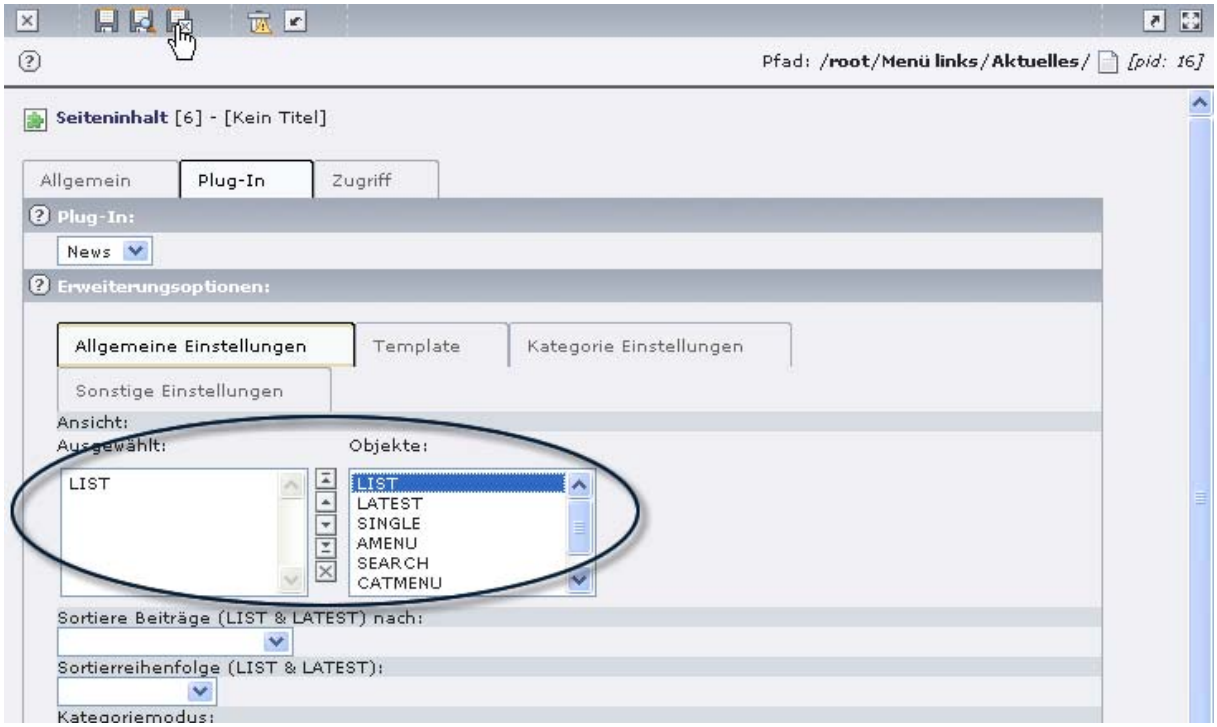
Die Maske wird sich umfangreich ändern, es stehen viele Optionen für die Konfiguration des News Plugins zur Verfügung.



Das Frontend Plugin tt_news ist sehr umfangreich, daher sollten Sie die Dokumentation zum Modul ausgiebig lesen. Wir werden nur eine der am häufigsten eingesetzten Grundkonfigurationen nutzen.

Im ersten Feld muss die gewünschte Ansicht ausgewählt werden. Die Ansicht LIST zeigt eine bestimmte Anzahl an News-Meldungen in einer Kurzfassung an. Die LATEST-Ansicht ist ähnlich der LIST-Ansicht, lässt jedoch andere Arten der Darstellung zu. SINGLE gibt an, dass nur ein einzelner Beitrag angezeigt werden soll, jedoch der gesamte Inhalt der News. Welche News angezeigt werden sollen, muss der SINGLE-Ansicht über eine Parameter-Übergabe mitgeteilt werden.

Wir werden eine Listen-Ansicht realisieren und wählen daher im Feld „Ansicht“ das Objekt „LIST“ aus. Speichern Sie anschließend den Datensatz.

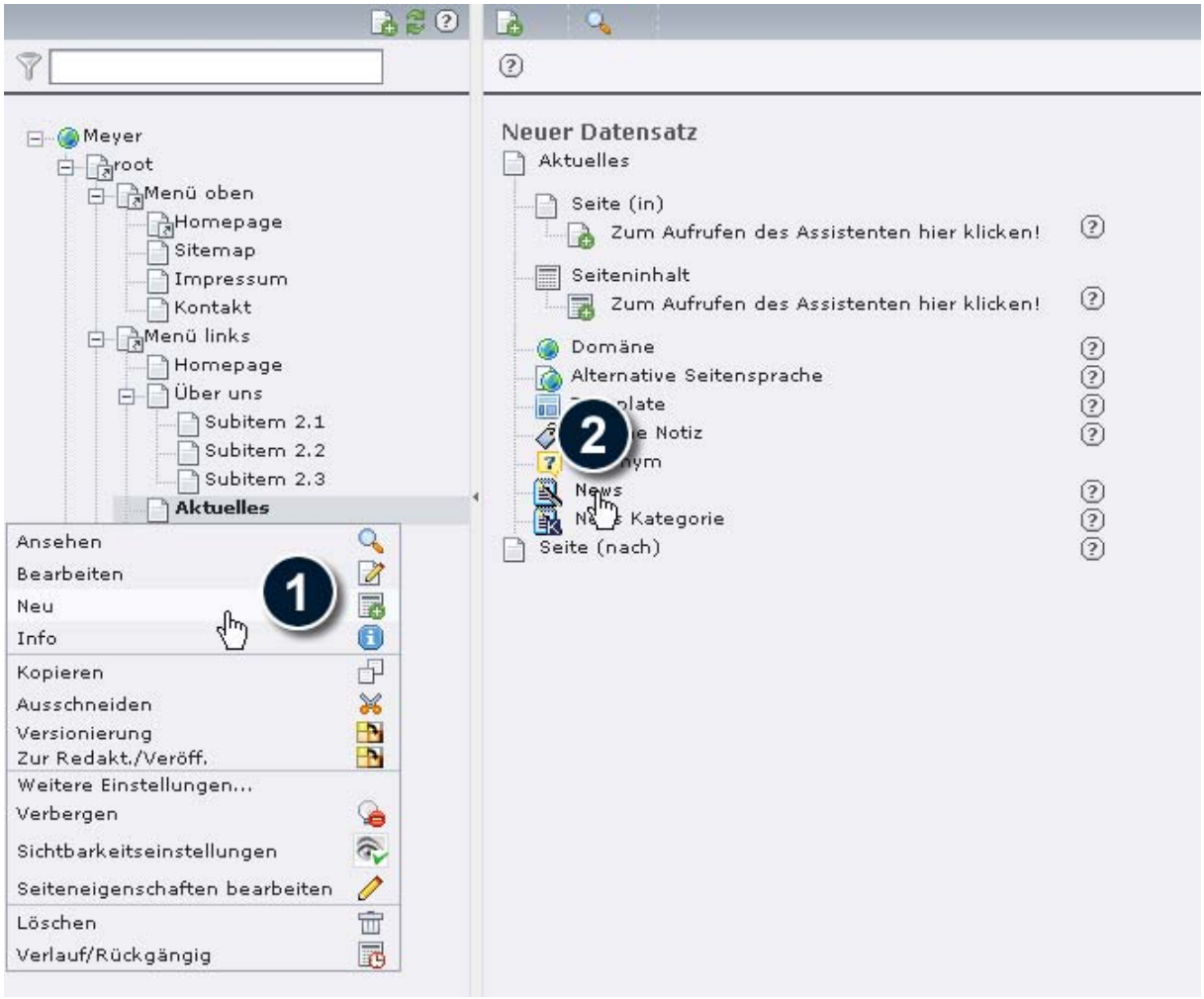


6.2.11 Mögliche Codes des News-Moduls

Ansicht	Beschreibung
LIST	Listet alle News auf.
LATEST	Listet die neusten, nicht archivierten News auf.
SINGLE	Zeigt eine bestimmte Nachricht vollständig an.
AMENU	Erzeugt ein Menü der archivierten News.
SEARCH	Suchfunktion innerhalb der News
CATMENU	Erzeugt ein Menü der News-Kategorien

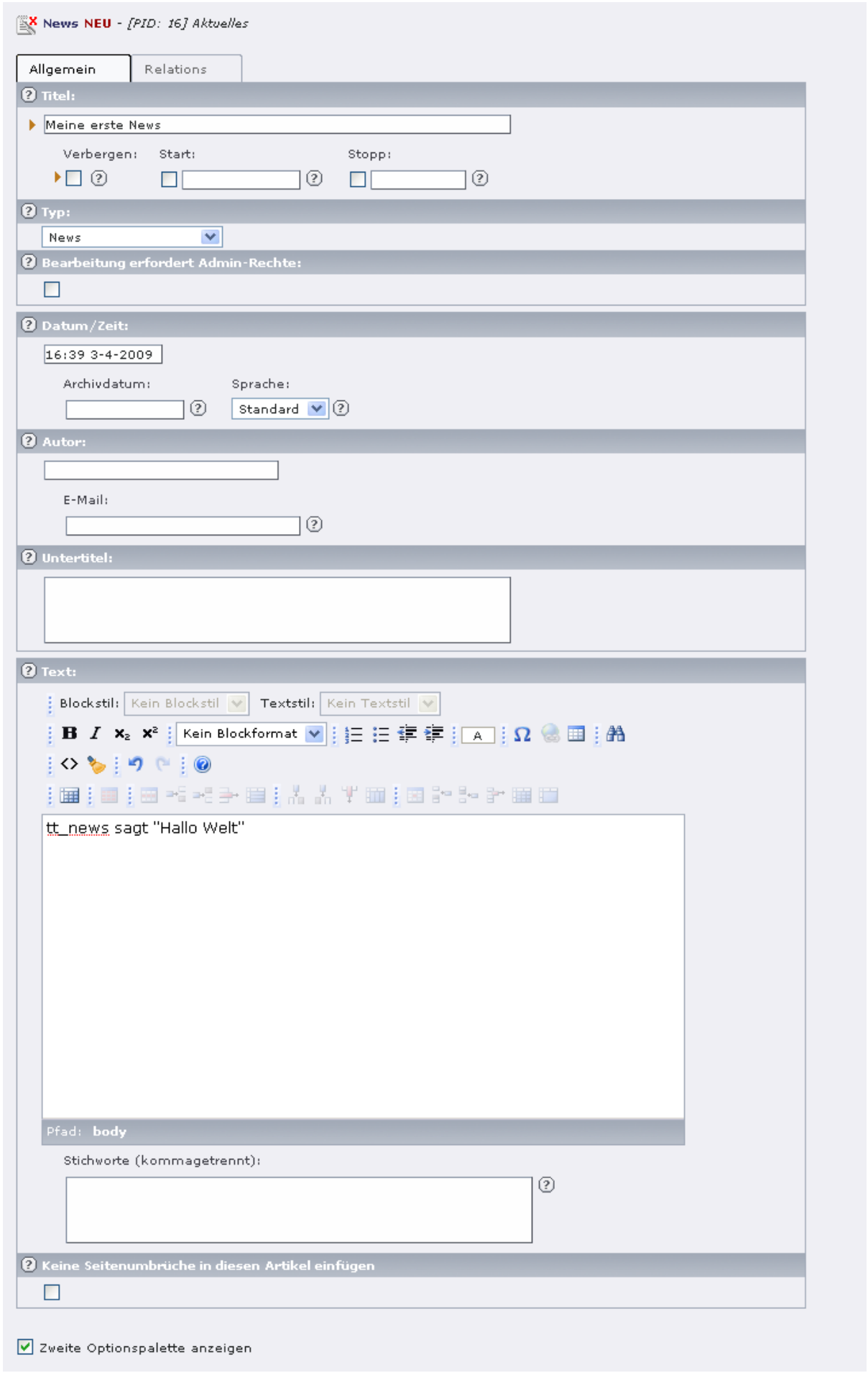
6.2.12 Newsbeiträge erstellen

Damit überhaupt News angezeigt werden können, müssen wir einige News anlegen. Legen wir nun auf unserer Seite "***Aktuelles***" einen neuen Datensatz an (keine neue Seite, keinen Seiteninhalt, sondern einen Datensatz). Wir klicken auf das Icon der Seite vor "***Aktuelles***" im Seitenbaum und wählen aus dem sich öffnenden Pop-up-Menü den Eintrag „***Neu***“. Wir sehen dann in der Liste der möglichen Datensätze den 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.

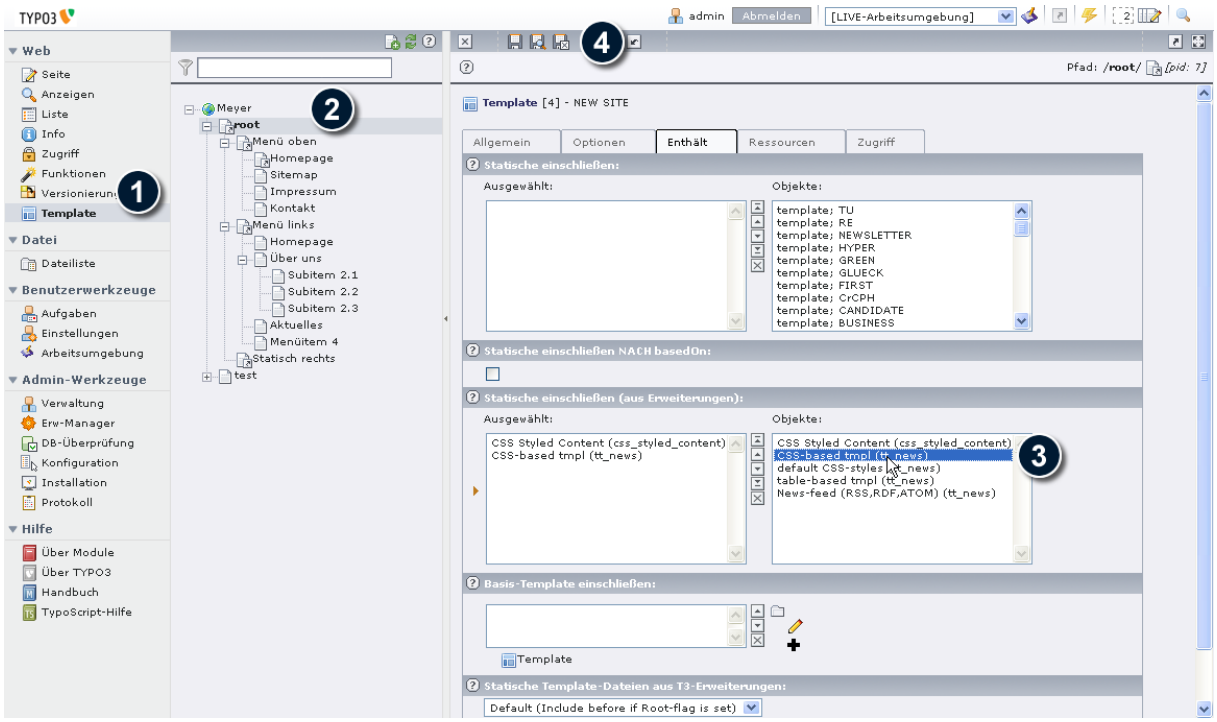
Tragen Sie in die Felder "***Titel***" und "***Text***" einen beliebigen Inhalt ein. Die Option "***Verstecken***" soll deaktiviert werden. Speichern und schließen Sie Ihren News-Datensatz.


News NEU - [PID: 16] Aktuelles
Allgemein
Relations
Titel:
Meine erste News
Verbergen:
Start:
Stopp:
Typ:
News
Bearbeitung erfordert Admin-Rechte:
Datum/Zeit:
16:39 3-4-2009
Archivdatum:
Sprache:
Standard
Autor:
E-Mail:
Untertitel:
Text:
Blockstil:
Kein Blockstil
Textstil:
Kein Textstil
Kein Blockformat
tt_news sagt "Hallo Welt"
Pfad: body
Stichworte (kommagetrennt):
Keine Seitenumbrüche in diesen Artikel einfügen
Zweite Optionspalette anzeigen

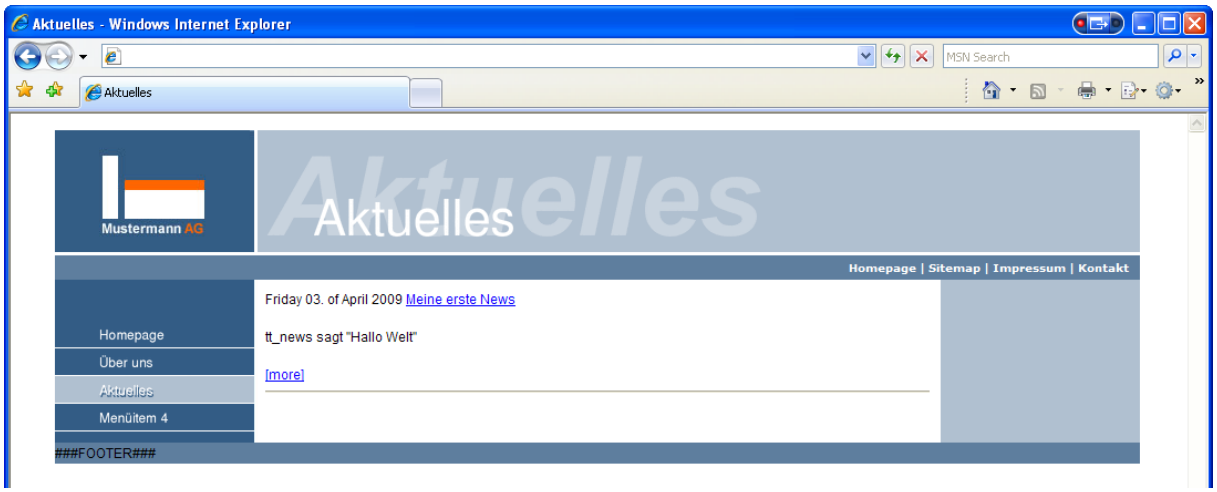
Wenn Sie die Seite „Aktuelles“ nun im Frontend betrachten, werden Sie feststellen, das keine News ausgegeben werden. Keine Panik, wir sind kurz vor dem Durchbruch, zum Anzeigen sind noch einige kleine Konfigurationen nötig.

6.2.13 Statische Templates für die Anzeige von News inkludieren

Damit News im Frontend angezeigt werden, muss ähnlich wie css_styled_content ein statisches Template im Root-Tenplate inkludiert werden. Wir wechseln dazu in das Modul „***Template***“ und klicken unsere Root Seite „***root***“ an, um das Root-Template zu bearbeiten. Wir klicken auf den Textlink „***Click here to edit the hole template record***“, um den gesamten Datensatz zu bearbeiten. In der bearbeiten Maske wechseln wir auf den Reiter „***Enthält***“ und fügen im Abschnitt „***Statische Einschließen (aus Erweiterungen)***“ das statische Template ***CSS-based tmpl (tt_news)*** ein. Wir speichern den Datensatz über das „Speichern und schließen“ Icon.



Wenn Sie nun das Frontend aufrufen, werden die News angezeigt. Das Design wird von dem Plugin vorgegeben, da dieses eine eigene Designvorlage mitbringt, die angepasst werden kann.

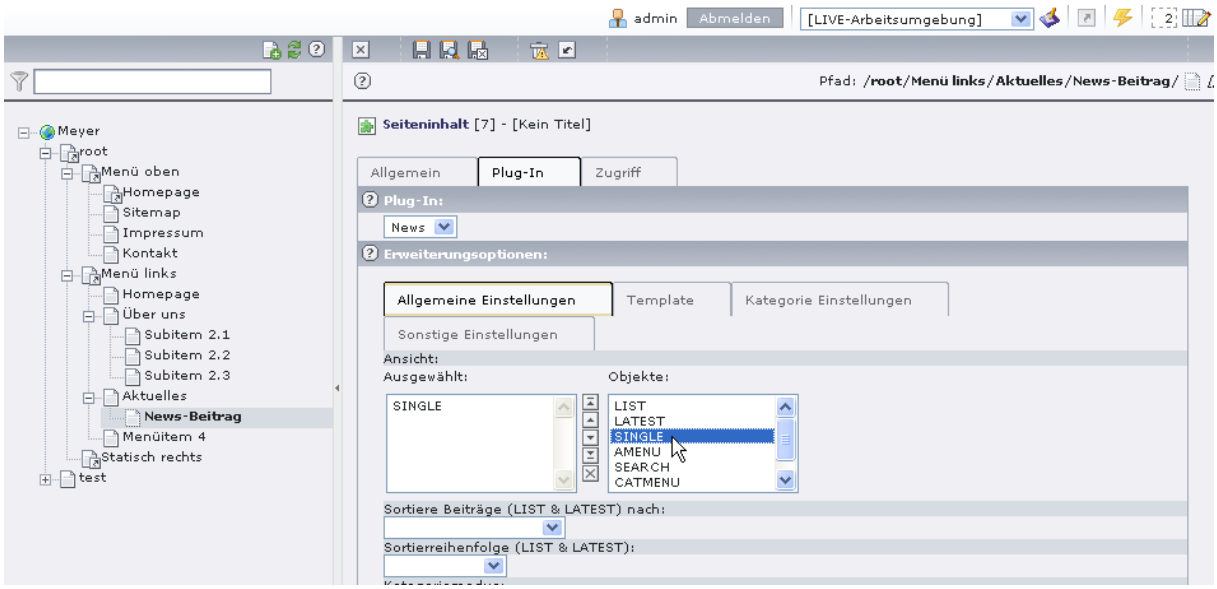


6.2.14 Die SINGLE-Ansicht

Wenn Sie im Frontend auf der Seite „Aktuelles“ auf einen News-Beitrag klicken, werden Sie wieder auf der gleichen Seite landen. Hier sollte jedoch „News“ in Ihrer Gesamtheit mit allen Informationen erscheinen. Im News-Plugin im Backend haben wir bisher nur die LIST-Ansicht konfiguriert.

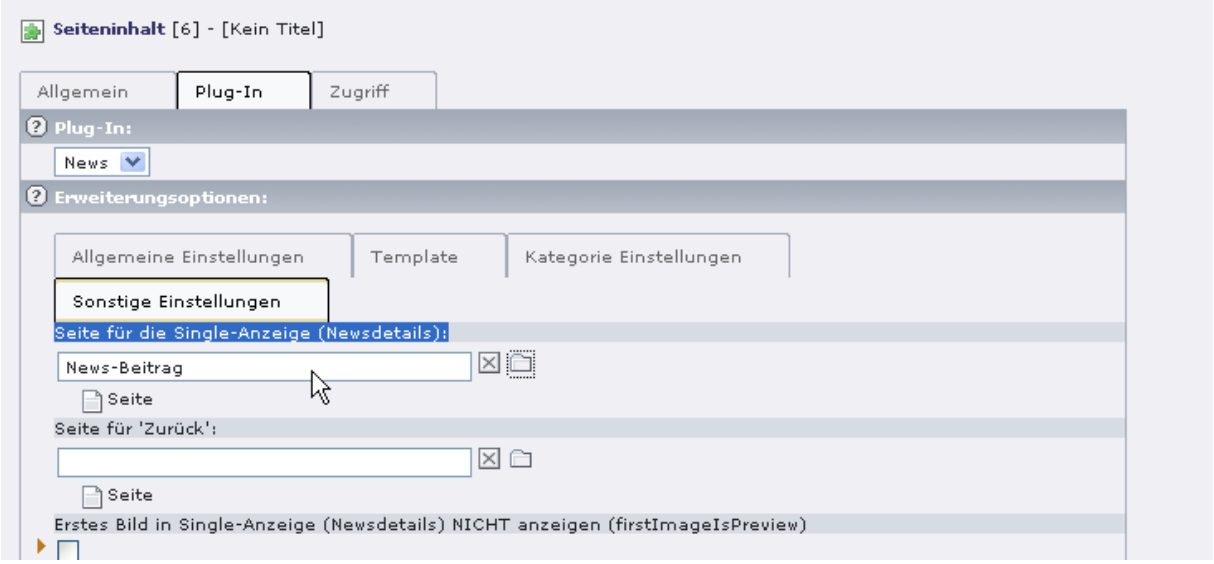
Wir legen für die SINGLE-Ansicht eine Unterseite der Seite „Aktuelles“ an und geben dieser den Titel „News-Beitrag“. In den Seiteneigenschaften aktivieren wir die Option „***Im Menü verbergen***“, da die Seite nicht im Menü erscheinen soll.

Auf der neuen Seite legen wir einen Seiteninhalt vom Typ „***Plugin einfügen***“ an. Wir wählen aus der Liste Plug-In den Eintrag „***News***“ aus, um das News Plugin als Seiteninhalt anzulegen. Im Feld „***Ansicht***“ wählen wir nun „***SINGLE***“ aus und speichern den Datensatz anschließend über die „Speichern und schließen“-Schaltfläche.



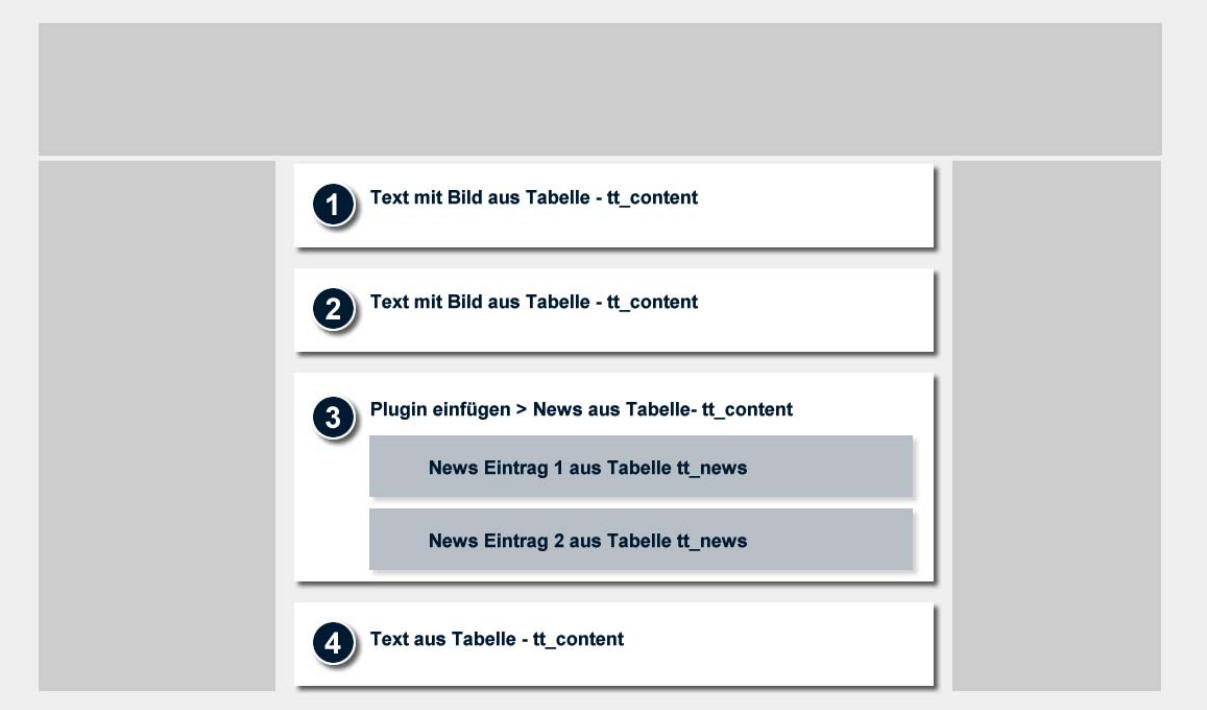
Um eine Verbindung der Listen-Ansicht zur Single-Ansicht zu erstellen, müssen wir dem Plugin noch mitteilen, wo sich die neu angelegte Seite mit der Single Ansicht befindet. Wir wählen im Seitenbaum die Seite „Aktuelles“ aus und klicken auf das Bleistiftsymbol des dort vorhandenen Seiteninhaltes, dem News Plugin.

Wir wechseln in den Reiter „Plugin“, in dem sich im Abschnitt „Erweiterungsoptionen:“ der Reiter „Sonstige Einstellungen“ befindet. Auf diesem Reiter können wir die Eigenschaft „Seite für die Single-Ansicht (Newsdetails):“ setzen, indem wir ganz rechts neben dem Eintrag auf das Icon „Datensätze durchblättern“ klicken. In dem sich öffnenden Fenster wählen wir die Seite mit unserer Single-Ansicht aus, also die Seite „News-Beitrag“ und speichern den Datensatz anschließend.



6.2.15 Frontend-Plugin: Elemente & Container

Das News-Modul ist ein klassisches Beispiel 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:



6.2.16 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? Und 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	Generelle Plugin Beschreibung, z.B Author, was das Plugin macht und wie es heißt, Version, Abhängigkeiten zu anderen Extensions und weitere Informationen zum Plugin.
ext_Tabels.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_Tabels.sql	Die Datei enthält den SQL-Dump, die Beschreibung von Tabellen und Felder der Datenbank. Bei der Installation des Moduls werden nach dieser Datei die Tabellen und Felder angelegt bzw. erweitert, die das News-Modul benötigt.
static/ts-new/setup.txt	In dieser Datei befindet sich der Standard Typoscript Code der

	inkludiert wird, wenn Sie im Template unter „Statische einschließen (aus Erweiterungen):“ das statische Template „CSS based Tmpl (tt_news) einbinden. Der Code lässt sich über den Template Analyser einsehen Es wird ermöglicht, das es eine Basiskonfiguration gibt und dass das News Plugin als Inhalt unterhalb von tt_content implementiert ist.
static/ts-new/constants.txt	Ähnlich wie static/ts-new/setup.txt jedoch enthält diese Datei-Variablen und die Beschreibung, wie der Constants Editor für das News Plugin auszusehen hat.
tca.php	Beschreibt wie die News Plugin Maske im Backend aussieht. Hier wird beschrieben, welche Felder in welcher Reihenfolge mit welcher Eigenschaft zur Verfügung stehen.
locallang_*.php	Sprachdateien, um das Backend mehrsprachig zu halten
pi/locallang.xml	Sprachdateien, um das Frontend mehrsprachig zu halten
pi/class.tx_tt_news.php	Die eigentliche PHP-Klasse, die die Funktionalitäten im Frontend bereitstellt.
pi/tt_news_v2_template.html	Die 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 tca.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 üblicherweise 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 und die temporären Caching-Dateien (temp_*) im Ordner "typo3conf/" 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 so genannte "Extension-Key" gespeichert.
pi_flexform	<?xml version="1.0" encoding="……	Enthält die Konfiguration der in den Registrierkarten angegebenen Werte im XML-Format.

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:

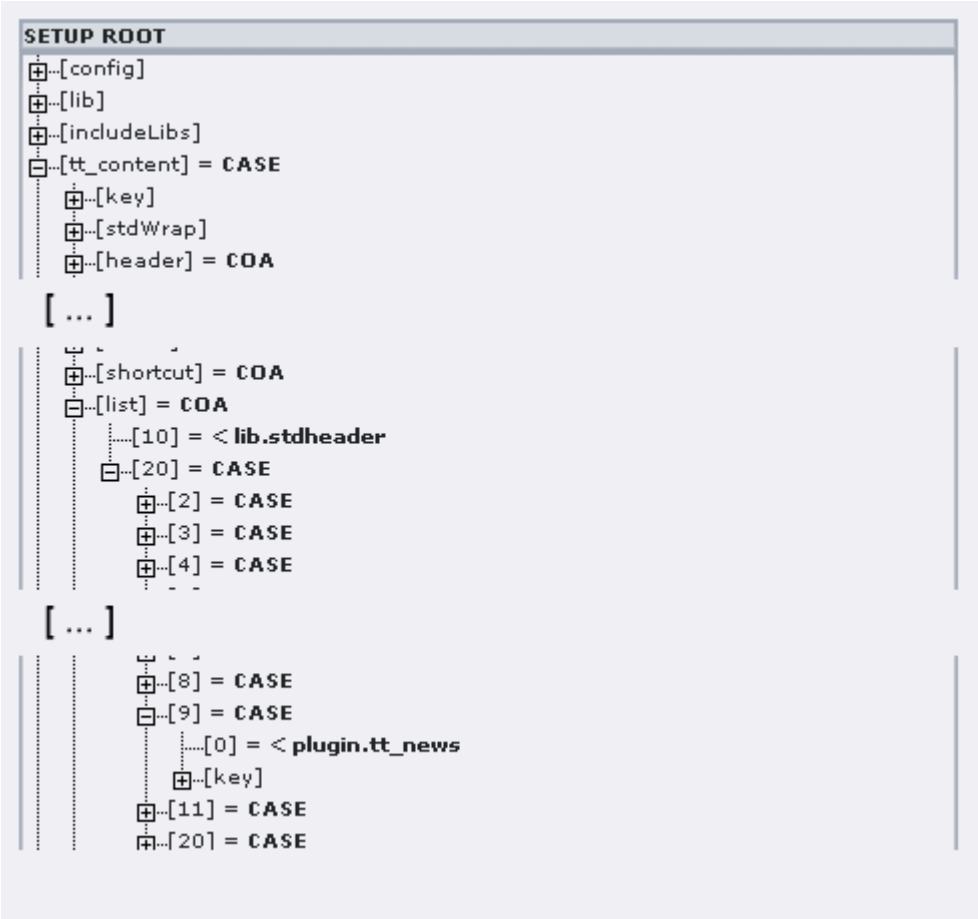
```
[...] marks.CONTENT = CONTENT
[...] marks.CONTENT.table = tt_content
```

Wie bereits im Abschnitt 3.9.2. erläutert, sucht TYPO3 für jeden gefundenen 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 Einbinden des statischen Templates für das News Plugin 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.



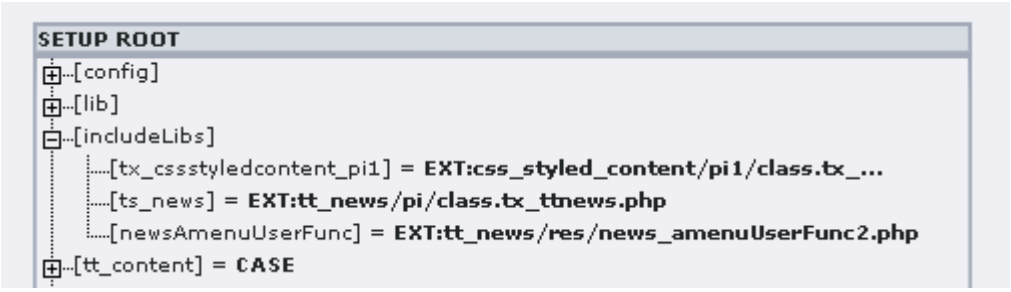
Zunächst geht TYPO3 in tt_content, um sich die Konfiguration für den aktuellen Datensatz zu suchen. tt_content ist mit "CASE" definiert, unter tt_content.key finden wir, auf welches Datenbankfeld die CASE-Abfrage ausgeführt wird, dies ist das Feld CType. Im Feld CType ist bei unserem Datensatz der Wert "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" referenziert.

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 Extension-Ordner (typo3/sysext, 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. Wesentlich ist, dass mehrere Klassen eingebunden werden können und hierzu eine weitere eindeutige Separation unterhalb von "includeLibs" notwendig ist.

6.2.17 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 "hart codiert" 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 wäre (Grundgedanke: "Install in one click"), werden Frontend-Module, wie z.B. das News-Modul, in drei Bereiche unterteilt: Funktionalität (php- Klasse), Konfiguration (Flexforms, TypoScript) und Design (Designvorlage).

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

6.2.18 Das News-Modul konfigurieren

Wie unter 6.2.17 erwähnt, sind Konfigurationen des News-Moduls über TypoScript möglich. Hierbei handelt es sich im Regelfall nicht direkt um das eigentliche 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.

Die über Flexforms, also direkt im Seiteninhalt angegebenen Konfigurationen, überschreiben im Regelfall die über TypoScript angegebenen Eigenschaften.

Wir haben im Abschnitt 6.2.16 gelernt, 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 möglichst an die Regeln halten, wie Eigenschaften in TypoScript geschrieben werden: Das erste Wort wird klein geschrieben, kommen mehrere Wörter in einer Eigenschaft vor, kennzeichnet man sie jeweils durch einen Großbuchstaben am Wortanfang, wodurch eine Eigenschaft besser lesbar wird. Die Eigenschaft bzw. Variable ***templateFile*** entspricht also dieser Konvention. Eigenschaften von tt_news können in einer Übersicht in der Dokumentation der Extension komplett eingesehen werden.

6.2.19 Die Designvorlage anpassen

Das Design kann relativ problemlos an die eigenen Bedürfnisse angepasst werden. Wichtig ist allerdings, dass Sie die Designvorlage nicht direkt im Extension-Ordner bearbeiten, da die Designvorlage ansonsten bei einem Update überschrieben würde.

Seit der TYPO3 Version 4.0 wird die Extension tt_news nicht mehr mit dem Source Code ausgeliefert, so dass sich die Extension nach dem Herunterladen aus dem Extension Repository in der Regel lokal im Verzeichnis typo3conf/ext/tt_news befindet.

Kopieren Sie sich die original Designvorlage aus dem Verzeichnis typo3conf/ext/tt_news/pi1/tt_news_v2_template.html 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
```

Wenn Sie die Template Datei über TypoScript festlegen, so wird diese als default-Wert bei jedem Einfügen eines tt_news Plugin-Seiteninhaltes verwendet. Innerhalb des Seiteninhaltes können Sie über ein Flexform-Feld eine Template-Datei angeben und beispielsweise eine zweite Designvorlage anlegen und diese nutzen. Die Flexform Konfiguration überschriebt so die TypoScript-Einstellung.

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.

Cache-Problem!
Sollten Sie Änderungen an Ihrer Designvorlage durchführen und diese nicht im Frontend angezeigt werden, löschen Sie stets den FE-Cache im Backend, bevor Sie auf Fehlersuche gehen!

6.2.20 Statische Darstellung von News

Mit folgendem TypoScript-Code ist es z.B. möglich, dass auf der rechten Seite permanent eine Latest Ansicht der News angezeigt wird:

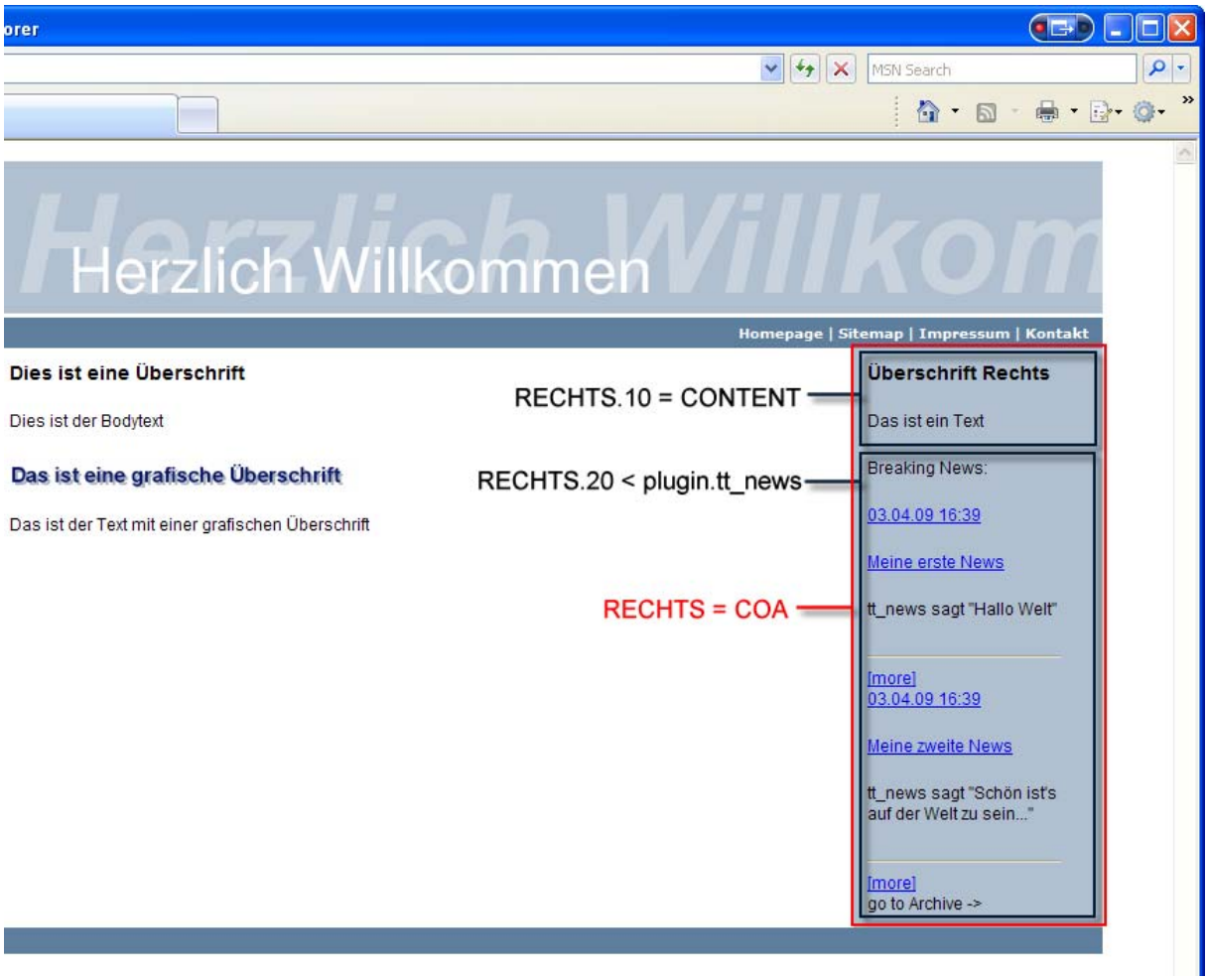
```
RECHTS < plugin.tt_news
```

```
RECHTS{
        code >
        code = LATEST
        pid_list >
        pid_list = 16
        singlePid >
        singlePid = 22
}
```

Da wir in unserem Root Template bereits Inhalt aus der rechten Spalte in den Marker RECHTS einfügen, müssen wir zuvor ein COA-Objekt einfügen, um so erst den Inhalt der rechten Spalte darzustellen und danach die Latest Ansicht der News:

```
 97   RECHTS = COA
 98   RECHTS.10 = CONTENT
 99   RECHTS.10 {
100         table = tt_content
101         select.orderBy = sorting
102         select.where = colPos = 2
103   }
104
105   RECHTS.20 < plugin.tt_news
106   RECHTS.20{
107         code >
108         code = LATEST
109         pid_list >
110         pid_list = 16
111         singlePid >
112         singlePid = 22
113   }
```

- In Zeile 97 weisen wir dem Marker RECHTS eine COA-Instanz zu.
- In Zeile 98 – 103 erzeugen wir ein CONTENT-Objekt in RECHTS.10 und stellen so den Inhalt der rechten Spalte dar.
- In Zeile 105 laden wir nach RECHTS.20 das Plugin tt_news
- In Zeile 107 löschen wir eventuell vorhandene Code-Eigenschaften und legen als Ansichts Code „LATEST“ fest
- In Zeile 109 – 112 übergeben wir die Seiten IDs für den Speicherort der News Datensätze (Zeile 110) und teilen dem News Plugin mit, auf welcher Seiten ID die Single-Ansicht zu finden ist (Zeile 112).


MSN Search
Herzlich Willkommen
Homepage | Sitemap | Impressum | Kontakt
Dies ist eine Überschrift
Dies ist der Bodytext
Das ist eine grafische Überschrift
Das ist der Text mit einer grafischen Überschrift
RECHTS.10 = CONTENT
RECHTS.20 < plugin.tt_news
RECHTS = COA
Überschrift Rechts
Das ist ein Text
Breaking News:
03.04.09 16:39
Meine erste News
tt_news sagt "Hallo Welt"
[more]
03.04.09 16:39
Meine zweite News
tt_news sagt "Schön ist's auf der Welt zu sein..."
[more]
go to Archive ->
Herzlich Willkom

7. Diverses

7.1 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."

Die Syntax von Conditions ist wie folgt:

```
[Gegebenheit1 = Wert1]&&[Gegebenheit2 = Wert2]
      TypoScript_A
[global]
```

Wenn Gegebenheit1 ***und*** Gegebenheit2 zutreffen, wird der Bereich "TypoScript_A" ausgeführt.

```
[Gegebenheit1 = Wert1] || [Gegebenheit2 = Wert2]
      TypoScript_A
[global]
```

Wenn Gegebenheit1 ***oder*** Gegebenheit2 zutrifft, wird der Bereich "TypoScript_A" ausgeführt.

Oder in Kombination:

```
[Gegebenheit1 = Wert1] || [Gegebenheit2 = Wert2] && [Gegebenheit3 = Wert3]
      TypoScript_A
[global]
```

Wenn Gegebenheit1 oder [***Gegebenheit2 und Gegebenheit3***] zutreffen, wird der Bereich "TypoScript_A" ausgeführt.

Eine vollständige Übersicht über die Möglichkeiten von Conditions finden Sie im Kapitel 7.1. 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.

Sollten Sie das Beispiel realisieren, so müssen Sie den Inhalt ungecached ausgeben, z.B. über ein Instanz eines COA_INT-Objektes.

Zum Testen setzten wir folgenden Code in unser Beispiel ein:

```
140   seite.10.marks.FOOTER = TEXT
141   [browser = msie]
142         seite.10.marks.FOOTER.value = Browser: Internet Explorer.
143   [browser = netscape]
144         seite.10.marks.FOOTER.value = Browser: Netscape/Mozilla.
145   [global]
```

Wenn Sie anschließend die Seite in unterschiedlichen Browsern (Internet Explorer und Netscape) aufrufen, so erhalten Sie jeweils eine unterschiedliche Anzeige.

Sie können Conditions auch im „TYPOScript Object Browser“ testen. Nachdem Sie Conditions angelegt haben, erscheinen diese im „Object Browser“. Aktivieren Sie diese und prüfen Sie das Ergebnis:

Löschen Sie die Condition nach dem Testen wieder, da wir den Marker FOOTER im folgenden Kapitel benötigen werden.

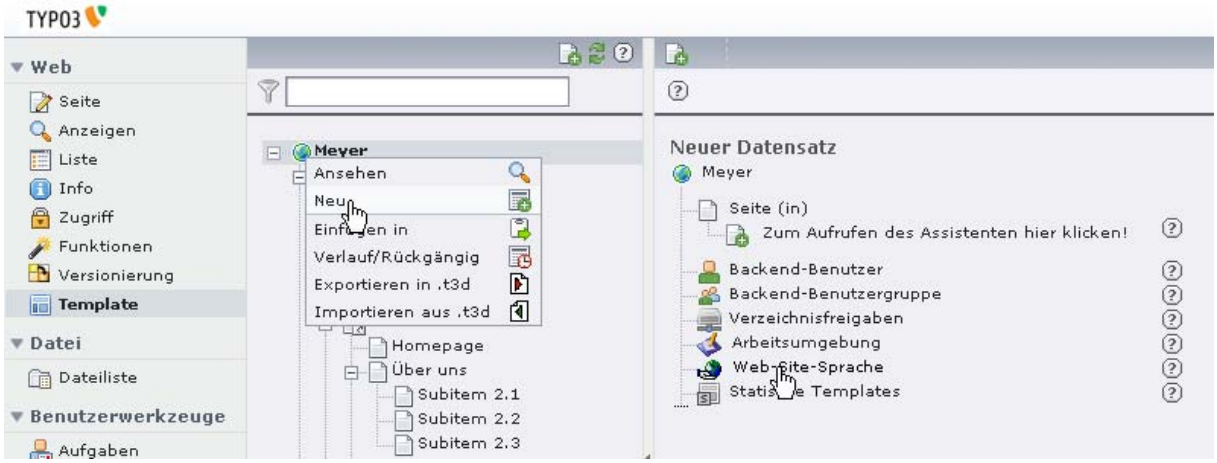
7.2 Mehrsprachige Webseiten

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.

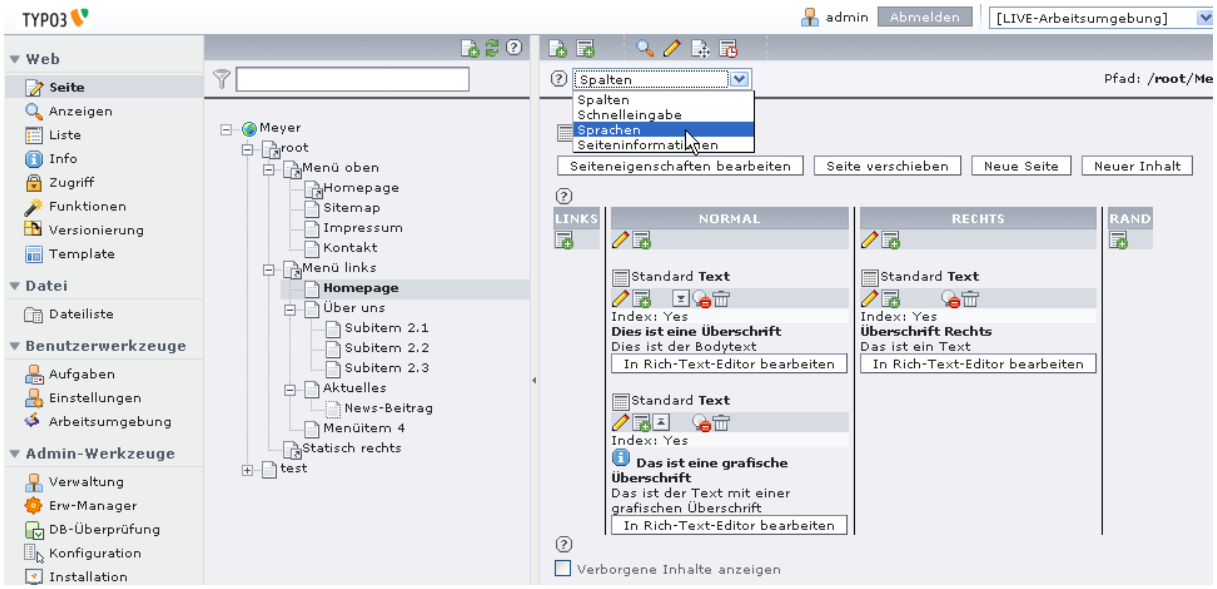
7.2.1 Seitensprachen, Seiten und Inhalte übersetzen

Zunächst müssen wir in TYPO3 eine weitere Sprache (zusätzlich zur bereits vorhandenen Default-Sprache) anlegen. Hierzu Icon vor der Rootebene, also auf die Weltkugel (nicht die Seite "root"!) und wählen „Neu“. Anschließend legen wir einen neuen Datensatz vom Typ "Web-Site-Sprache" an.



Geben Sie als Sprache z.B. "Englisch" an, wählen Sie die Länderflagge aus (gb.gif) und speichern Sie Ihren neuen Datensatz.

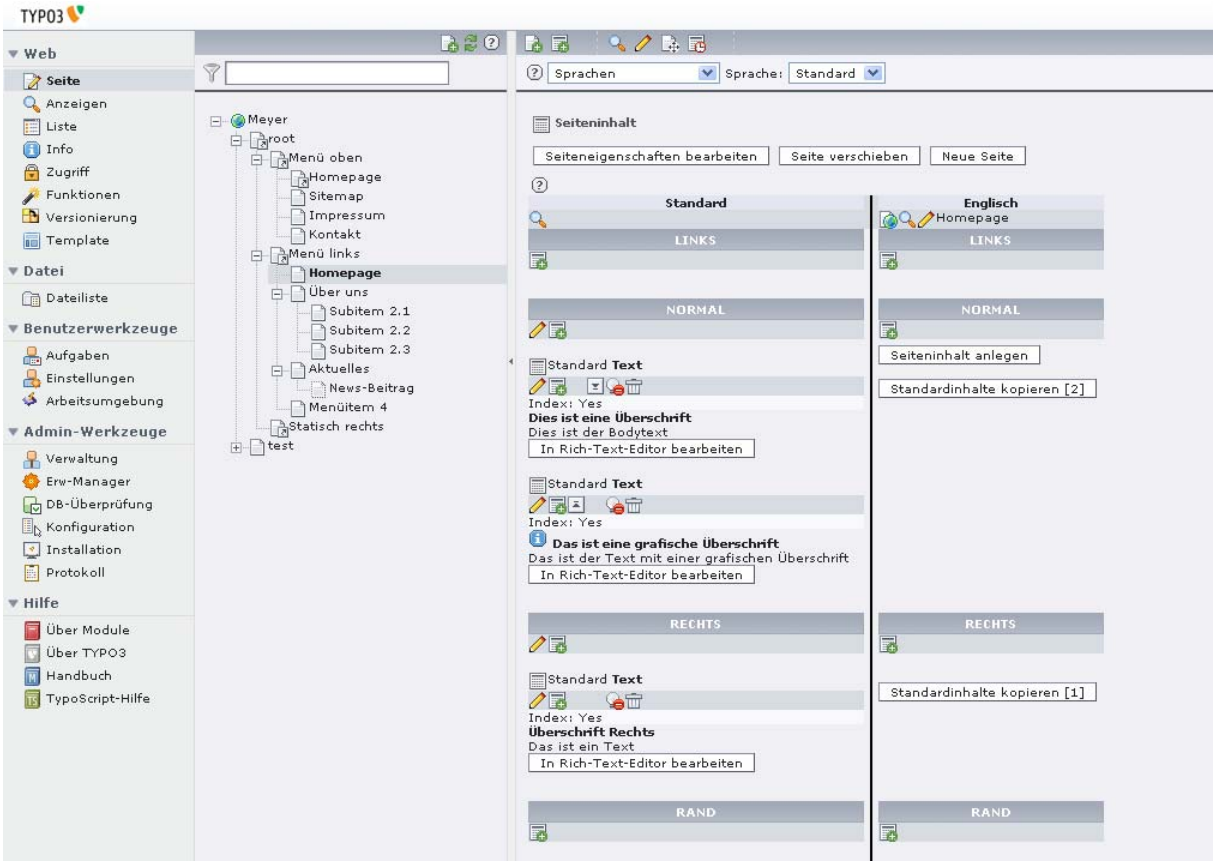
Nun haben wir eine neue Website-Sprache angelegt und müssen unsere Seiten und deren Inhalte entsprechend übersetzten. Hierzu gehen wir wieder in den Bereich "***Seite***" und wählen beispielsweise unsere Seite "***Homepage***" aus, auf der sich bereits Inhalt befindet. Wir können nun im oberen, rechten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



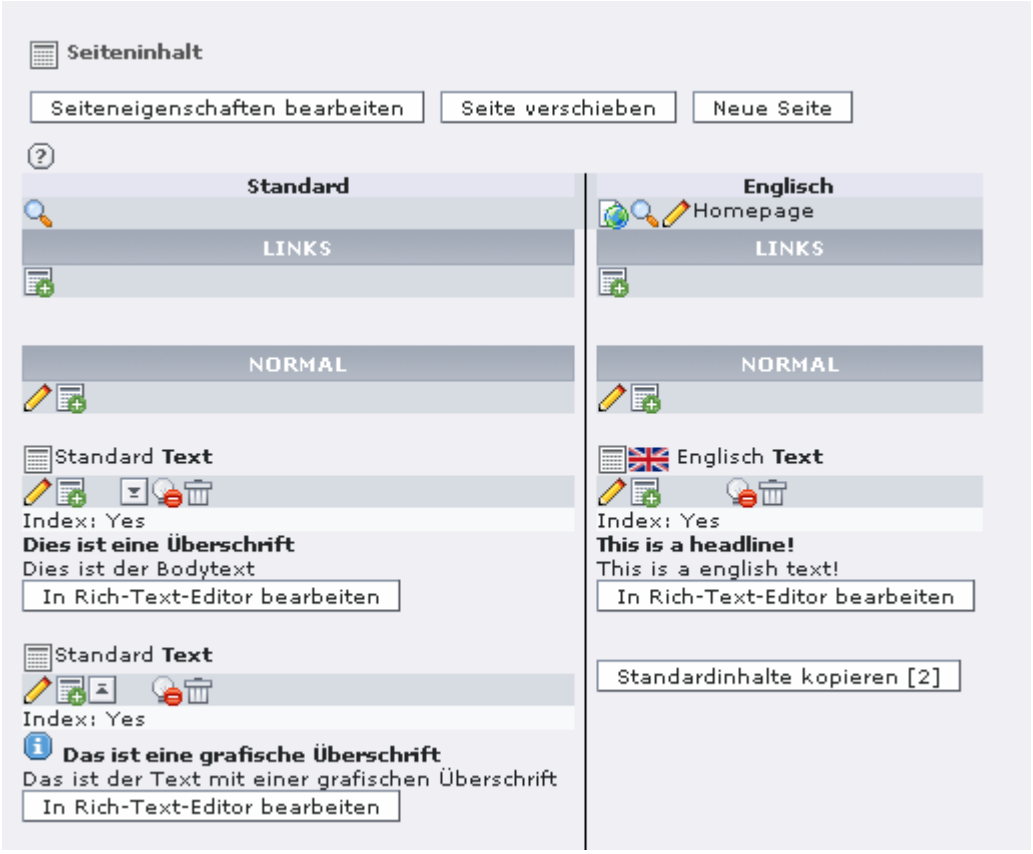
Das Element "***Sprachen***" steht nur im Modul „***Seite***“ zur Verfügung, und nur wenn auch eine zusätzliche Website-Sprache angelegt wurde!

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 ebenfalls "Homepage" sein). Achten Sie darauf, dass die übersetzte Version nicht als "Versteckt" gekennzeichnet ist und speichern Sie Ihre Übersetzung des Seitentitels.



Sie können nun die bereits existierenden Inhalte über die Schaltfläche „Standardinhalte kopieren [x]“ kopieren, um diese zu übersetzen, oder Sie legen in der Spalte "Englisch" nun einen neuen Seiteninhalt für die Übersetzung des Seiteninhalts an. Geben Sie als Überschrift "This is a Headline" vom Typ "This is an english text!" an und speichern Sie diesen neuen Seiteninhalt. Die Darstellung im Backend sollte nun wie folgt aussehen:



7.2.2 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 seltensten 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 deutschen Sprache dargestellt werden, müssen wir in TypoScript unsere Konfiguration für den Marker "MITTE_CONTENT" und „RECHTS“ erweitern:

```
 90   MITTE_CONTENT = CONTENT
 91         MITTE_CONTENT {
 92               table = tt_content
 93               select.orderBy = sorting
 94               select.where = colPos = 0
 95               select.languageField = sys_language_uid
 96         }
 97
 98         RECHTS = COA
 99         RECHTS.10 = CONTENT
100         RECHTS.10 {
101               table = tt_content
102               select.orderBy = sorting
103               select.where = colPos = 2
104               select.languageField = sys_language_uid
105         }
```

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, möglichst oberhalb von "seite = PAGE":

```
01 config.linkVars = L
02 config.uniqueLinkVars = 1
03 config.sys_language_uid = 0
04 config.language = de
05 config.localeAll = de_DE
06
```

Und fügen unterhalb unseres gesamten TypoScriptes, also am Ende, folgende Condition ein:

```
148 [globalVar = GP:L=1]
149   config.sys_language_uid = 1
150   config.language = en
151   config.localeAll = gb_UK
152 [global]
```

- In Zeile 1 wird angegeben, dass ein Parameter „L“ mit jedem von TYPO3 erzeugten Link weitergegeben wird.
- In Zeile 2 wird über config.linkVars festgelegt, das ein Parameter nur einmalig in der URL vorkommen darf.
- In Zeile 3 und 4 wird die ID der Sprache auf 0 /default Sprache) und das Language Label auf de gesetzt.
- In Zeile 5 werden systemweite lokale Informationen, wie das Datum, auf die gewünschte Sprache gesetzt.
- In Zeile 148 wird eine Condition gesetzt, so dass die Zeilen 149 -151 nur ausgeführt werden, wenn der Parameter L = 1, also die englische Sprache, aktiviert ist. In Zeile 149 - 151 werden dann alle Spracheinstellungen auf Englisch gesetzt.

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

7.2.3 Einen einfachen Sprachwechsel einrichten

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

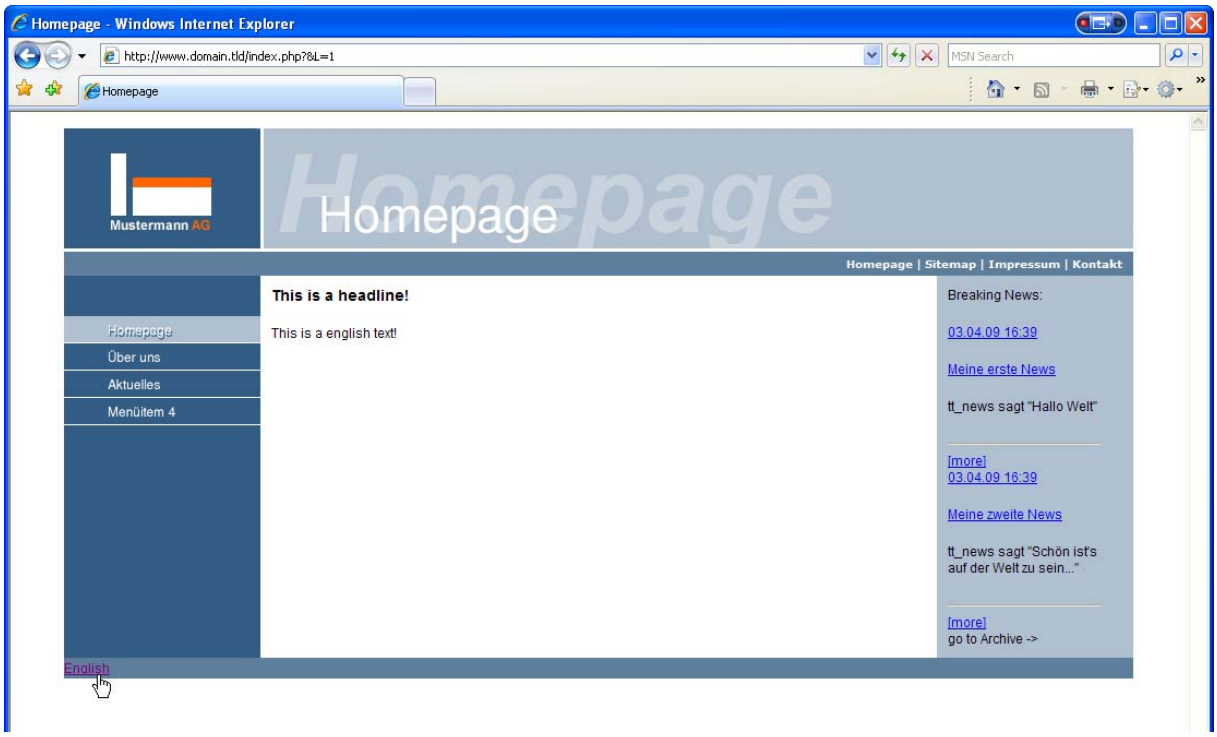
Wir werden Sprachwechsler in unseren Marker FOOTER integrieren, in der Regel sollte die Umschaltung jedoch im Kopfbereich einer Internetseite erscheinen. Prüfen Sie bitte, dass die Condition aus dem Kapitel 7.1 den Marker nicht beschreibt.

Wir fügen nun folgenden TypoScript Code zu unserem Root Template hinzu:

```
 13 seite = PAGE
 14 seite {
      [...]

 24   10.marks{
120         RECHTS.20{
121         [...]
127         }
128
129         FOOTER = TEXT
130         FOOTER{
131               value = English
132               typolink.parameter.data = page:uid
133               typolink.additionalParams = &L=1
134         }
135   }
136 }
```

- Wir erzeugen im Marker FOOTER eine TEXT Instanz und weisen dieser den Wert „***English***“ zu (Zeile 129 -131).
- Mit der Funktion „***typolink***“ erzeugen wir den Link bzw. bauen diesen zusammen. Es wird ein Link benötigt, der auf die aktuelle Siete zurückführt, diese jedoch in englischer Sprache anzeigt. Über die Funktion „***parameter.data***“ wird die Tabelle ***page*** und die Spalte ***uid*** ausgelesen und zurückgeliefert (Zeile 132).
- Mit der Funktion additionalParameter wird nun dem Parameter „***&L=1***“ angehängt und somit auf die Ansicht der englischen Sprache umgeschaltet.



Der Textlink muss sich jedoch in Abhängigkeit der genutzten Sprache ändern, daher fügen wir folgende Änderungen in die Condition am Ende unseres TypoScriptes ein:

```
 07 seite = PAGE
 08 seite{
    [...]
130 }
    [...]
155 [globalVar = GP:L=1]
156   config.sys_language_uid = 1
157   config.language = en
158   config.localeAll = gb_UK
159
160   seite.10.marks.FOOTER.value = Deutsch
161   seite.10.marks.FOOTER.typolink.additionalParams = &L=0
162 [global]
```

- In Zeile 160 und 161 wird der Text auf „Deutsch“ geändert und der Parameter „***&L = 0***“ für die deutsche Sprache gesetzt. Die Änderungen werden nur ausgeführt, wenn die Sprache auf Englisch stand, also der Parameter *&L=1* in der URL übergeben wird.

7.3 Druckerfreundliche Version

Häufiger Anwendungsfall bei Internetpräsentationen ist eine spezielle 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 4.2.1 haben wir bereits die Eigenschaft "typeNum" des PAGE-Objektes kennen gelernt. Und genau diese

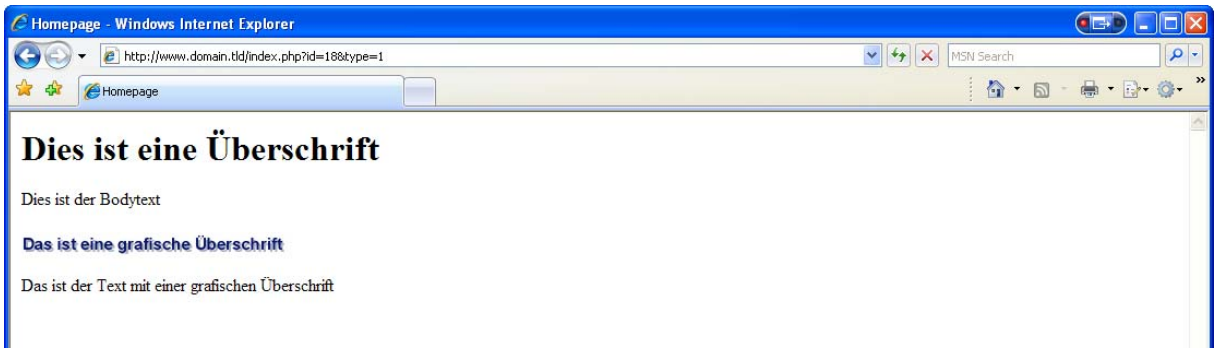
typeNum-Eigenschaft können wir uns nun für die Druckversion zunutze machen. Wir erweitern unser Template um folgende Zeilen:

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

Achten Sie darauf, dass die Erzeugung der Druckversion im TypoScript außerhalb der „seite“ geschieht, also nicht innerhalb der Klammerung des „seite“-Objektes!

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:

```
z.B. http://www.domain.tld/index.php?id=18&type=1
```



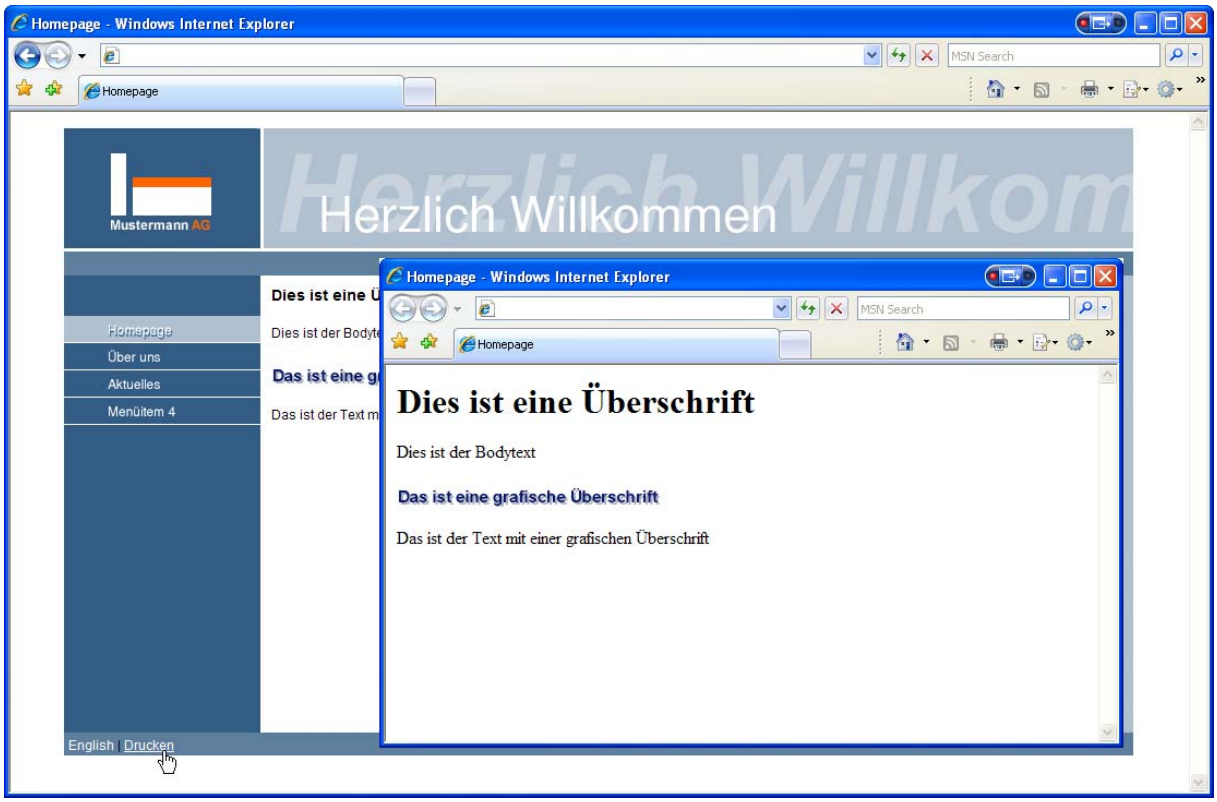
Selbstverständlich kann für die Druckversion auch eine Designvorlage verwendet werden. Hierzu ist es erforderlich, das Objekt TEMPLATE zu verwenden, also z.B.

```
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.

```
07 seite = PAGE
08 seite{
    [...]
123   FOOTER = COA
124   FOOTER.10 = TEXT
125   FOOTER.10{
126         value = English
127         typolink.parameter.data = page:uid
128         typolink.additionalParams = &L=1
129   }
130   FOOTER.20 = TEXT
131   FOOTER.20{
132         wrap =  &#124; |
133         value = Drucken
134         typolink.parameter.dataWrap = {TSFE:id},1
135         typolink.target = blank
136   }
137 }
```

- Die Funktion „***typolink***“ kennen wir schon vom Sprachwechsler, jedoch wird hier „***.parameter.datawrap***“ zur Link-Erzeugung verwendet. Es wird die Seiten ID ausgelesen und „type = 1“ gesetzt (Zeile 134).
- Damit sich die Druckansicht in einem neuen Fenster öffnet, setzen wir noch „***.target=blank***“ (Zeile 135).



Da wir den Footer verändert haben, müssen wir auch unsere Condition für den Sprachwechsler noch anpassen, damit die Funktion des Umschaltens weiter gegeben ist:

```
162 [...]
163 [globalVar = GP:L=1]
164   config.sys_language_uid = 1
165   config.language = en
166   config.localeAll = gb_UK
167
168   seite.10.marks.FOOTER.10.value = Deutsch
169   seite.10.marks.FOOTER.10.typolink.additionalParams = &L=0
170 [global]
171 [...]
```

7.4 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 Druckversion, sodass die grafische Überschrift, die wir im Abschnitt 5.4.21 konfiguriert haben, in der Druckversion nicht als Grafik, sondern im gleichen Format wie die Überschrift „Layout 1“ angezeigt wird:. Fügen wir die folgende Condition am Ende unseres Root Templates ein:

```
[globalVar = TSFE:type=1]
      lib.stdheader.10.5 < lib.stdheader.10.1
[global]
```

Bitte beachten Sie, dass Conditions sich nicht(!) innerhalb von geschweiften Klammern befinden dürfen!

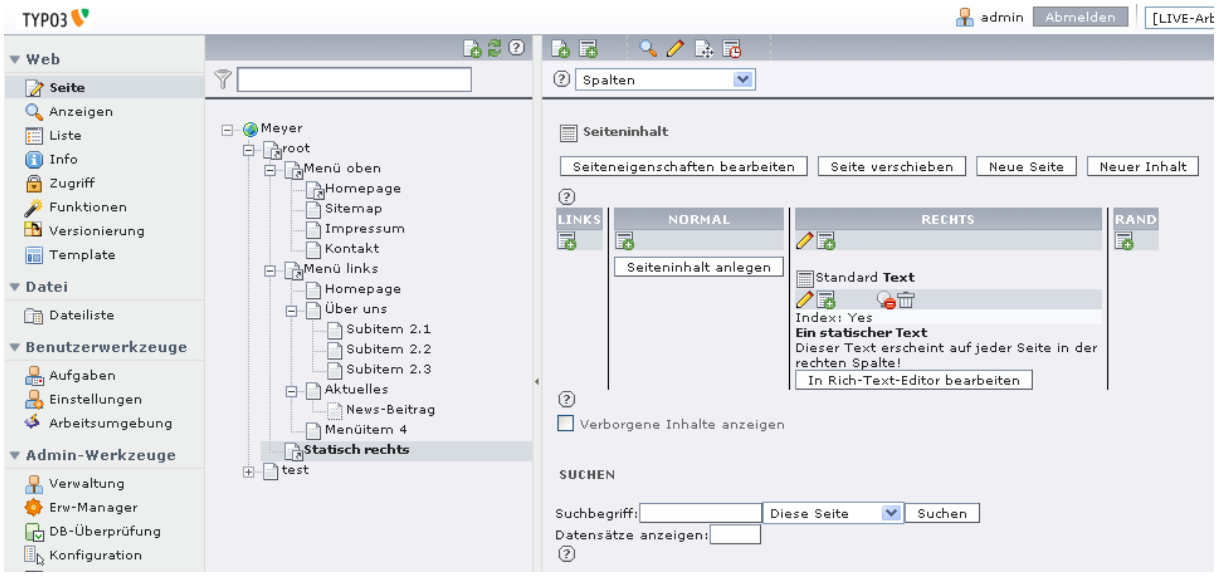
7.5 Statischer Content in der rechten Spalte

In der rechten Spalte sollen neben den Inhalten, die auf den Seiten in der rechten Spalte angelegt werden, auch Inhalte der Seite statisch Rechts permanent und auf jeder Seite zu sehen sein. Der statische Inhalt soll unter unserer LATEST News-Ansicht dargestellt werden:

```
 07 seite = PAGE
 08 seite{
    [...]
 22   10.marks{
            [...]
108         RECHTS = COA
109         RECHTS.10 = CONTENT
110         RECHTS.10{
111         [...]
115         }
116
117         RECHTS.20 < plugin.tt_news
118         [...]
119
120         RECHTS.30 = CONTENT
121         RECHTS.30{
122               table = tt_content
123               select.orderBy = sorting
124               select.where = colPos = 2
125               select.languageField = sys_language_uid
126               select.pidInList = 8
127         }
128    }
129 }
```

- In den Zeilen 120 -125 erstellen wir eine Instanz des CONTENT Objektes und lesen den Inhalt sortiert und nach Sprache geordnet aus der Tabelle tt_content aus, an der colPos 2, also aus der rechten Spalte
- Interessant ist die Zeile 126, in der wir festlegen, mit pidInList eine Liste von Seiten-IDs angeben zu können, von dem der Inhalt dargestellt werden soll. In unserem Fall ist dies die ID 8 der Site „statisch Rechts“

Wir legen nun in der Spalte „Rechts“ auf der Seite „statisch Rechts“ ein Textelement an:



Im Frontend wird nun in der rechten Spalte auf jeder Seite das Element angezeigt, das Sie auf der Seite „statisch Rechts“ angelegt haben.

7.6 Suchmaschinenfreundliche URLs

Suchmaschinen wie google „freuen“ sich in der Regel über Internetseiten mit URLs der Art „*.html“, da diese besser indiziert und von Suchmaschinen-Robotern durchsucht werden können.

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.

7.6.1 Den Webserver und TYPO3 vorbereiten

Um in den Genuss von SimulateStaticDocuments zu kommen, muss der Webserver wie Rewrite Regeln zulassen. Beim Webserver Apache muss das Modul „mod_rewrite“ verfügbar sein. Anschließend ist es zwingend erforderlich, eine .htaccess-Datei auf dem Webserver zu hinterlegen, die den folgenden Inhalt hat:

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

Hierdurch werden vom Webserver Seiten wie *.html an die Datei index.php geleitet, obwohl die Dateien physikalisch nicht existieren.

Wir müssen TYPO3 selbst noch beibringen, das es Seiten in der Form homepage.html erzeugen soll (statt index.php?id=18).

Wir fügen folgende Zeilen in unser Root-Template an erster Stelle ein, um simulateStaticDoccuments zu aktivieren:

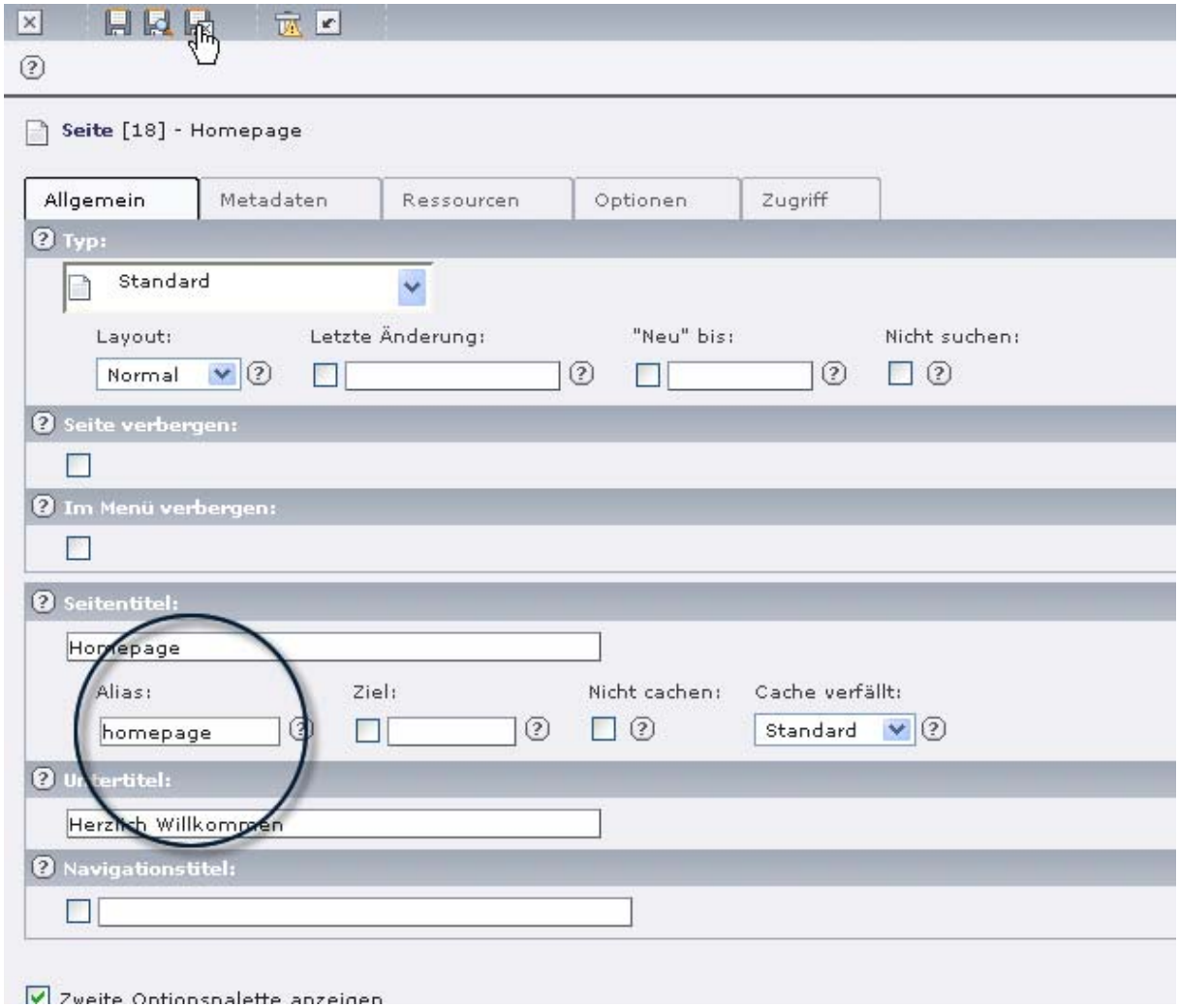
```
01 config.simulateStaticDocuments = 1
```

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".

Bei Kunden von Mittwald CM Service sind die in diesem Abschnitt genannten Einstellungen bereits voreingestellt.

7.6.2 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, dass 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
```

7.7 Benutzerrechte Backend-Redakteure

TYPO3 besitzt eine mächtige Rechteverwaltung, die es ermöglicht, Benutzer und Benutzergruppen mit ganz unterschiedlichen Berechtigungen anzulegen. Bei einer Standard-Installation von TYPO3 ist im Normalfall bereits ein Benutzer mit Administratorrechten angelegt. Mit diesem Benutzer haben wir bisher gearbeitet.

Häufig werden Sie Benutzer anlegen wollen, die nur auf bestimmte Seiten Zugriff haben und bestimmte Aktionen ausführen dürfen. Dazu können Backend-Benutzer, so genannte „Redakteure“, angelegt werden.

Das Rechte-Konzept basiert auf Benutzergruppen und den Benutzern selbst. Ein Benutzer, der kein Administrator ist, hat zunächst keine Rechte und darf nicht mehr als sich an- und abmelden. Die Rechte sollten generell über eine Benutzergruppe vergeben werden

Ein Benutzer kann Mitglied mehrerer Gruppen sein, die Rechte addieren sich. Hierbei arbeitet TYPO3 mit so genannten Positivlisten. Nur wenn eine Rechte-Eigenschaft 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.

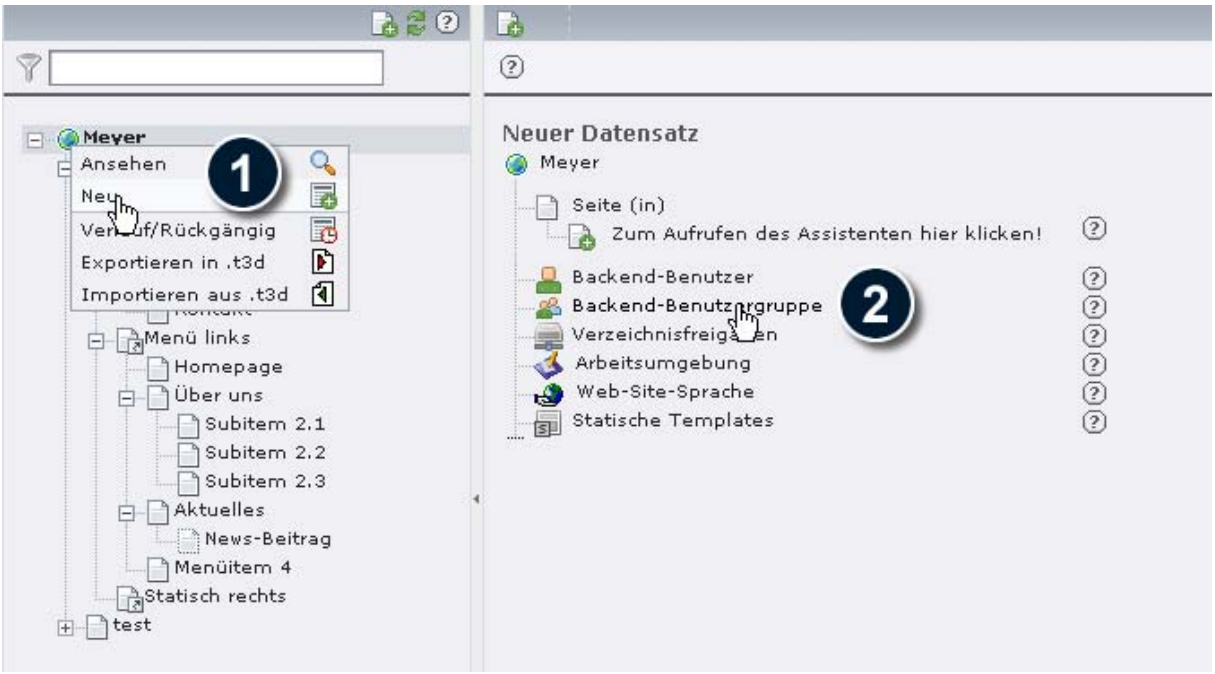
Wichtig zu wissen ist, dass für eine richtige Benutzerverwaltung an vier Stellen gearbeitet werden muss:

- Es muss eine Benutzergruppe geben, in der über Zugriffs-Listen (access lists) angegeben wird, auf welche Tabellen die Mitglieder der Gruppe zugreifen (lesen, schreiben) dürfen.
- Dann muss es selbstverständlich einen Benutzer geben, dessen Eigenschaften in der Regel, Benutzername, Passwort, Adressdaten und Gruppenzugehörigkeit sind.

- Es muss einen Datenbank- und Filemount geben, dabei handelt es sich um die Einstiegspunkte im Seitenbaum und die freigegebene Ordner in der Dateiliste für einen Upload von Dateien im fileadmin/Verzeichnis.
- Und, oftmals vergessen, müssen die Zugriffrechte für die entsprechenden Seiten explizit gesetzt werden.

7.8 Benutzergruppe anlegen

Wir legen eine neue Benutzergruppe an, indem wir auf der Root-Ebene (nicht der Seite „root“) auf das Symbol der Weltkugel klicken und „Neu“ auswählen. Wir legen einen neuen Datensatz vom Typ „Backend-Benutzergruppe“ an.

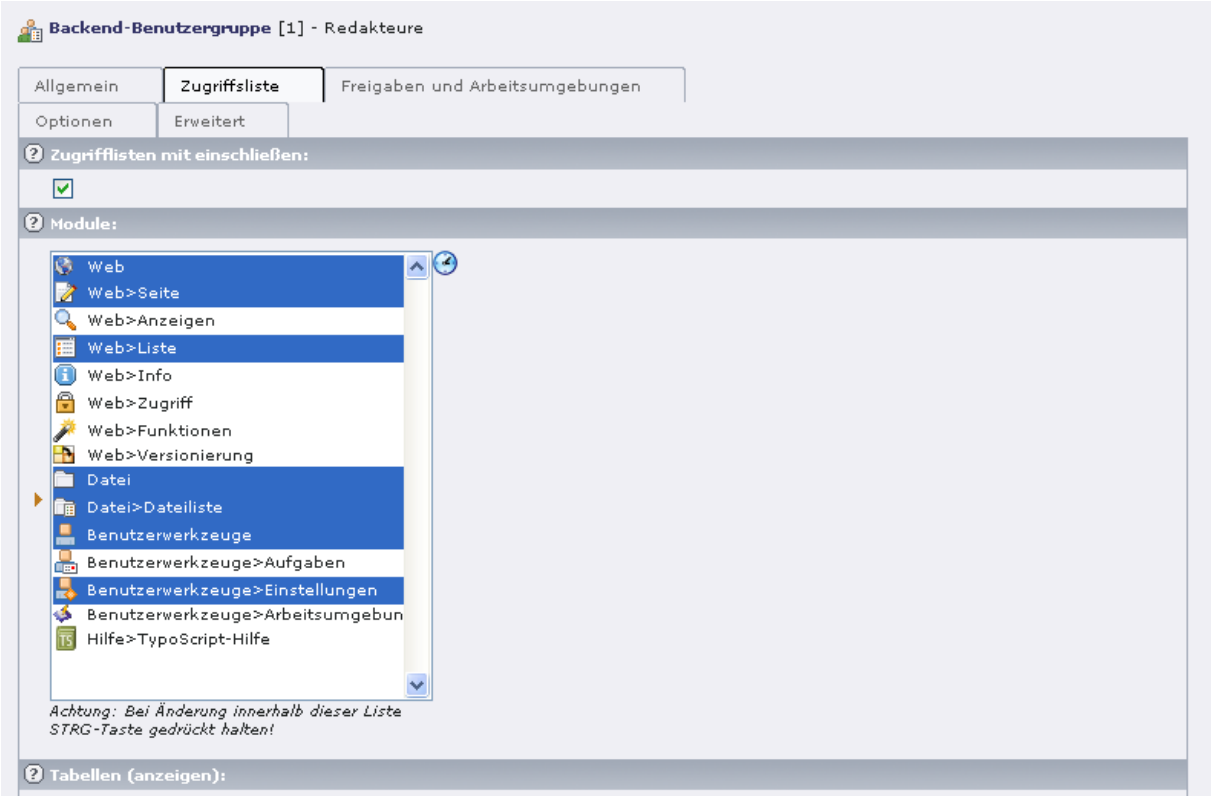


In der sich öffnenden Bearbeitungs-Maske der Benutzergruppe vergeben wir einen Gruppennamen „Redakteure“. Anschließend wechseln wir auf den Reiter „***Zugriffsliste***“ und aktivieren die Liste, indem wir den Haken bei „***Zugrifflisten mit einschließen***:“ setzen und die darauf folgenden Meldung mit OK bestätigen. Wir erhalten eine umfangreiche Liste mit Konfigurationsmöglichkeiten, in der wir angeben können, welche Backend-Module dem Redakteur zur Verfügung stehen sollen. Sinnvolle Module für Redakteure sind:

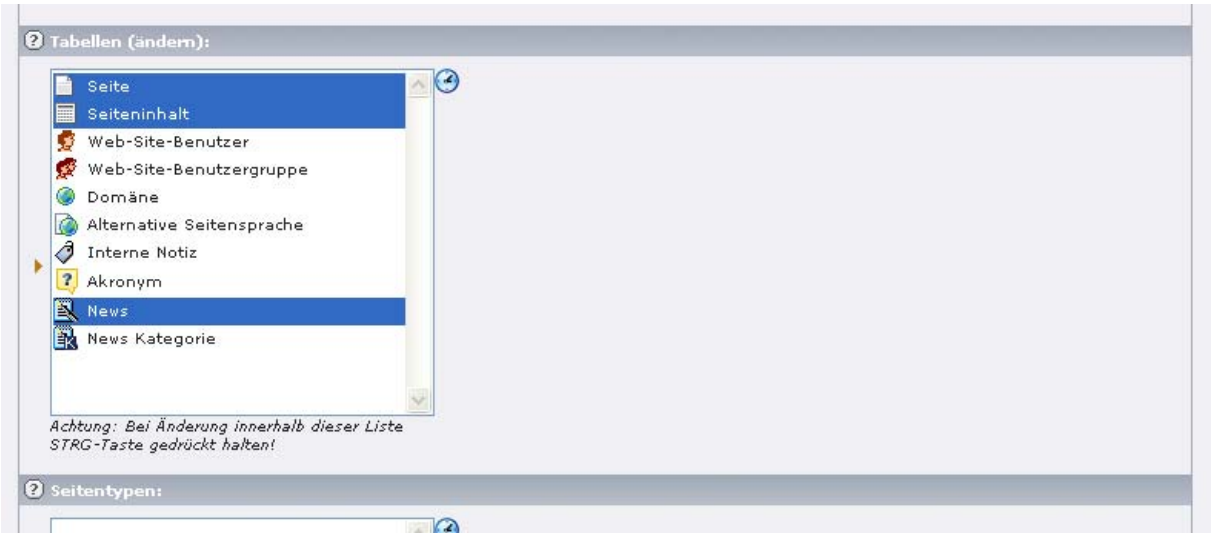
- **Web**
- Web>Seite
- Web>liste
- **Datei**
- Datei>Dateiliste
- **Benutzerwerkzeuge**

- Benutzerwerkzeuge>Einstellungen

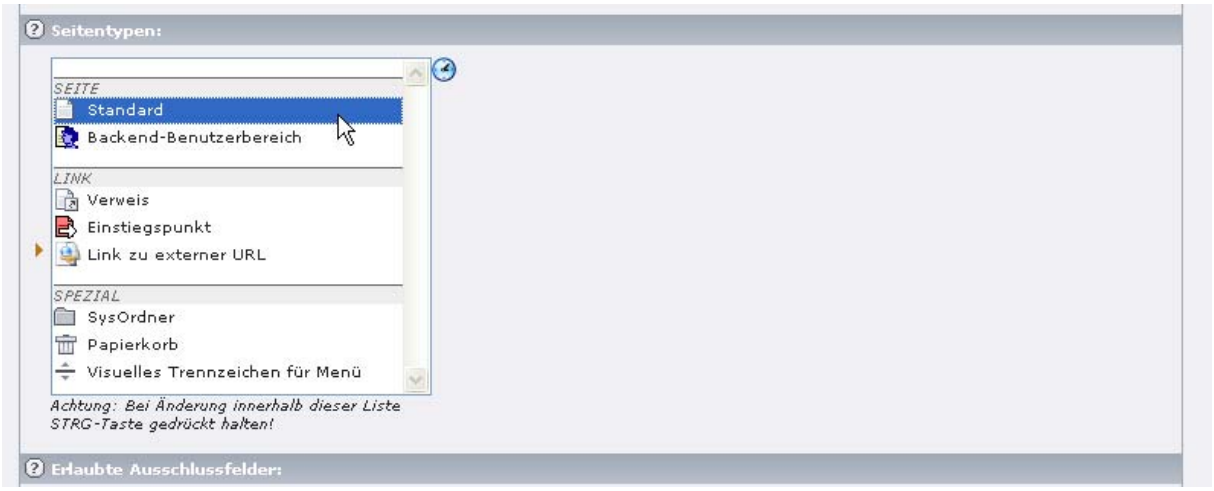
Eine Mehrfachauswahl können Sie durch Drücken der STRG-Taste ausführen, so das die folgenden Rechte ausgewählt sind:



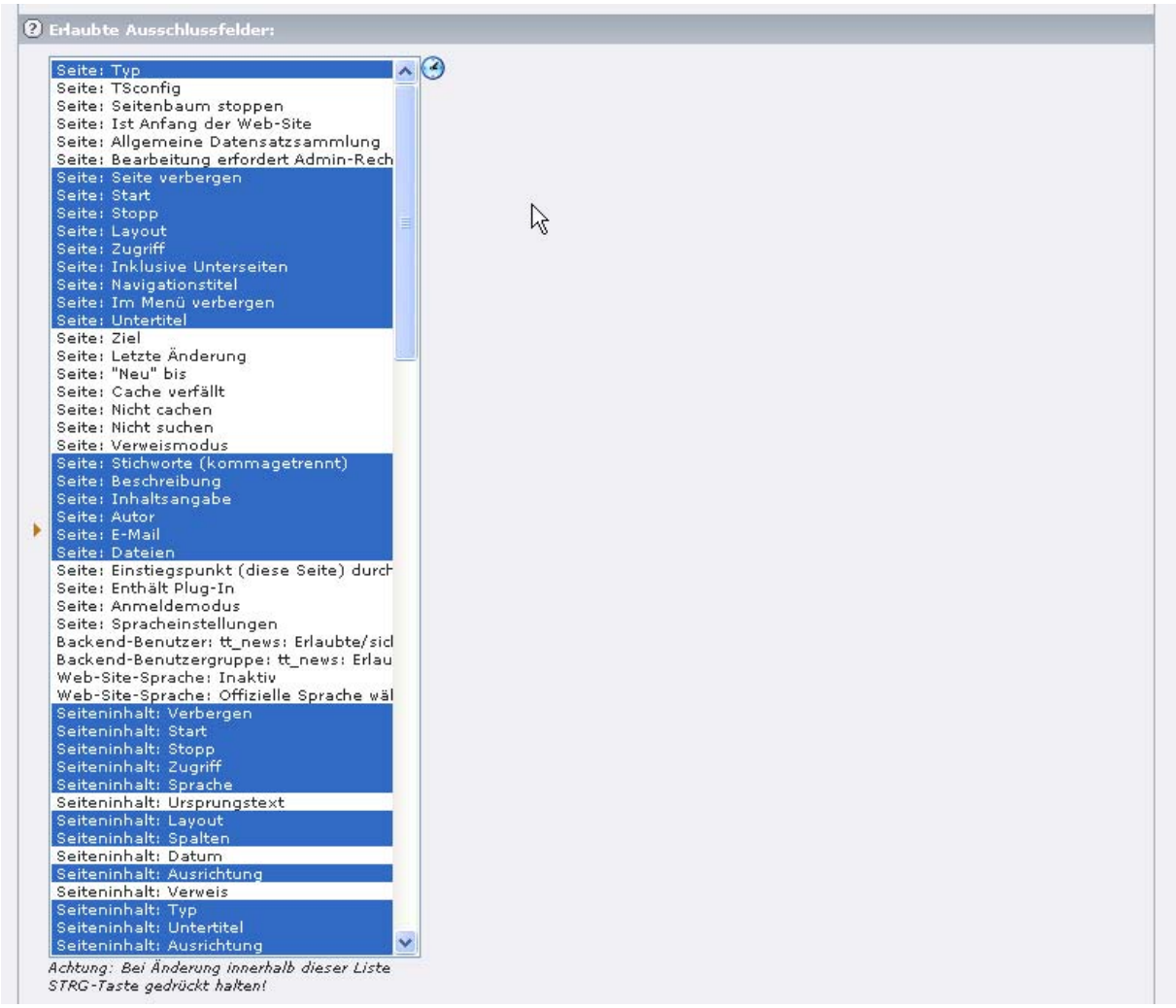
Als nächstes legen wir die Rechte für die Tabellen fest, welche der Benutzer sehen bzw. ändern darf. In der Liste werden auch installierte Erweiterungen, wie z.B. news, angezeigt. Unsere Redakteure sollen Seiten, Seiteninhalte und News anlegen können, also wählen wir bei Tabellen (ändern) diese aus:



Wir haben den Redakteuren nun erlaubt, Seiten und Seiteninhalte anzulegen. Wir müssen nun noch festlegen, welche Seitentypen angelegt werden dürfen und welche Felder für die Seiteninhalte zur Verfügung stehen. Im Feld Seitentypen erlauben wir nur den Seitentyp „Standard“:

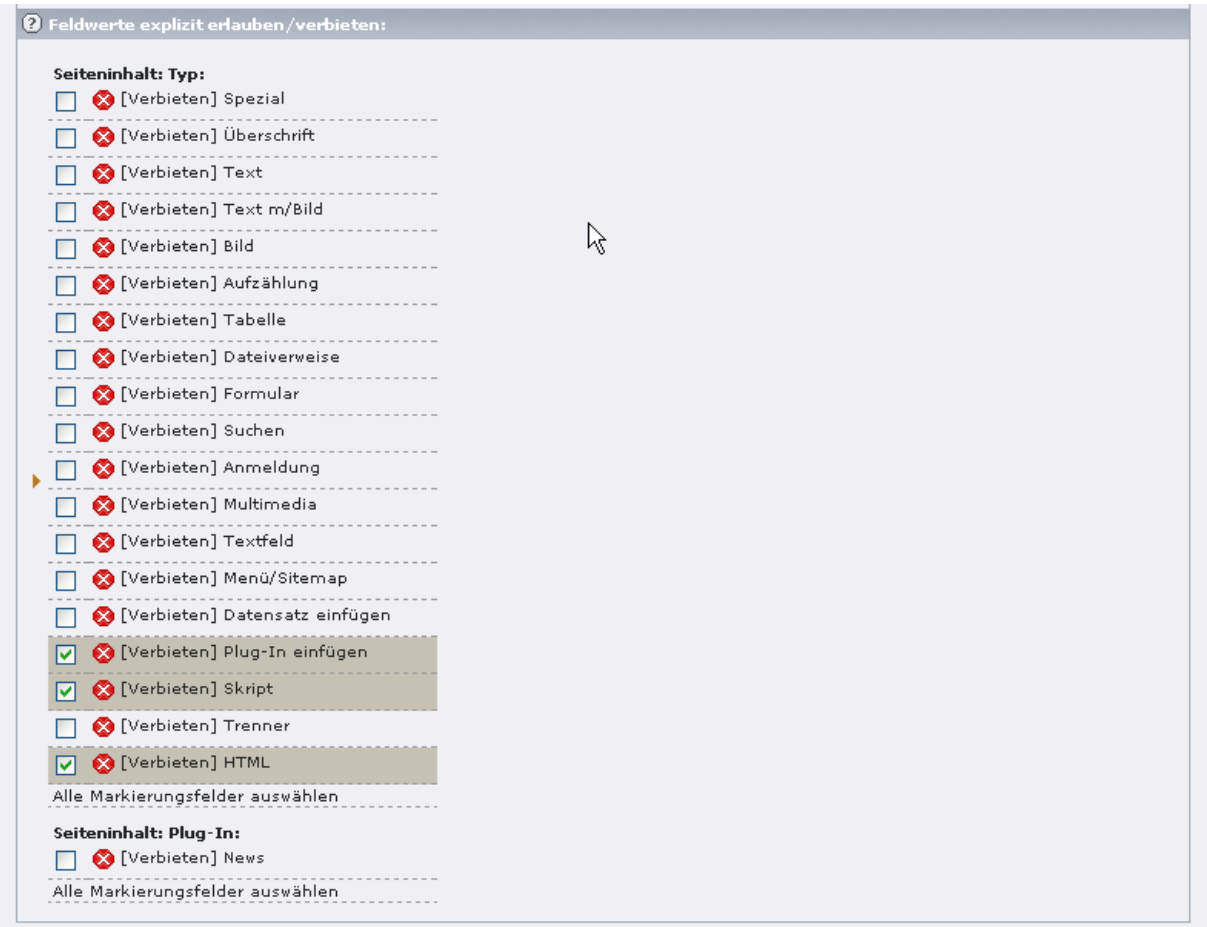


Es folgt eine umfangreiche Box „Erlaubte Ausschlussfelder“ mit sehr vielen Einträgen. Hier geben wir an, welche Felder ein Benutzer sehen kann. Übliche Einstellungen vom Typ Seite sind Typ, Seite verbergen, Start, Stopp, Zugriff, Dateien. So könnte eine Konfiguration aussehen:



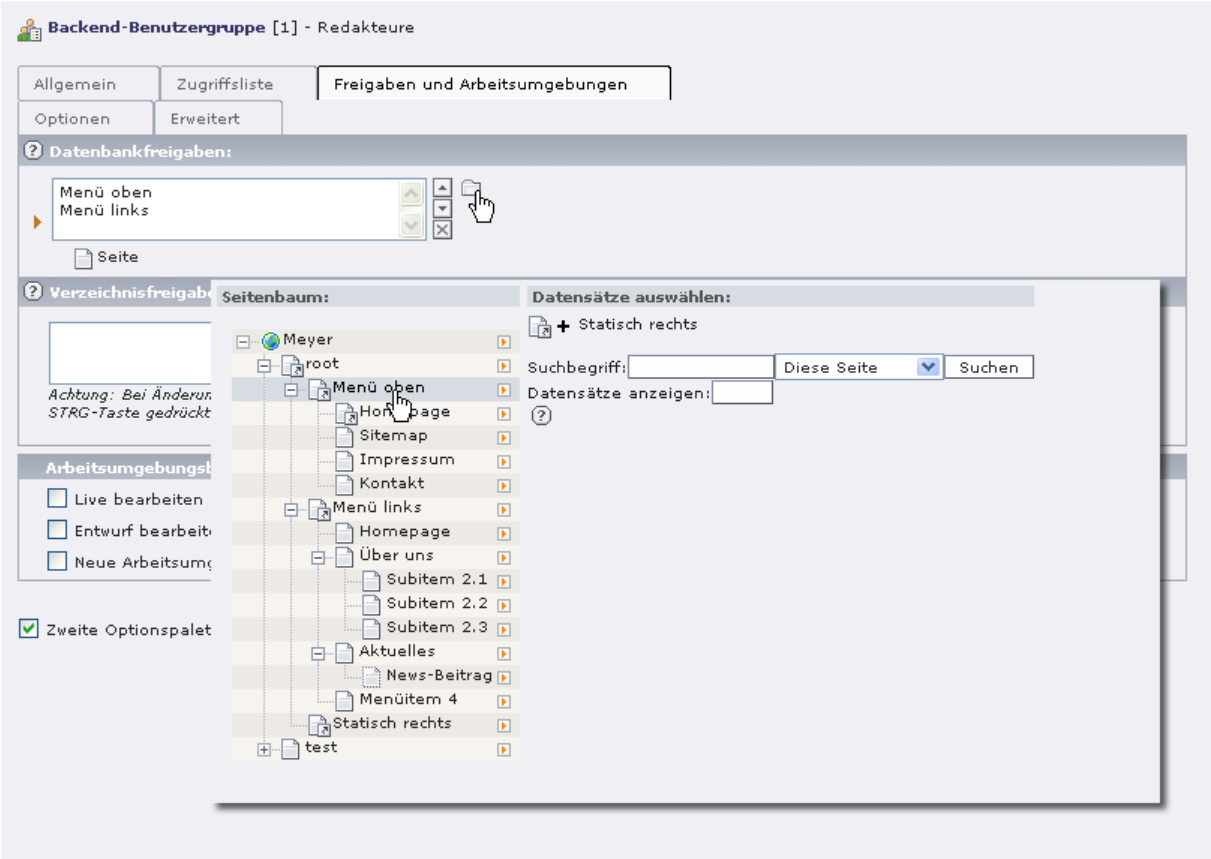
Die Rechte müssen Sie Ihren Bedürfnissen und den in TYPO3 genutzten Funktionalitäten anpassen.

Im Feld „Feldwerte explizit erlauben/verbieten:“ können wir angeben, welche Seiteninhaltstypen dem Redakteur **nicht (!)** zur Verfügung stehen. Zum Beispiel sollte er kein Skript und kein reines HTML eingeben können, dies liegt jedoch in Ihrer Hand:



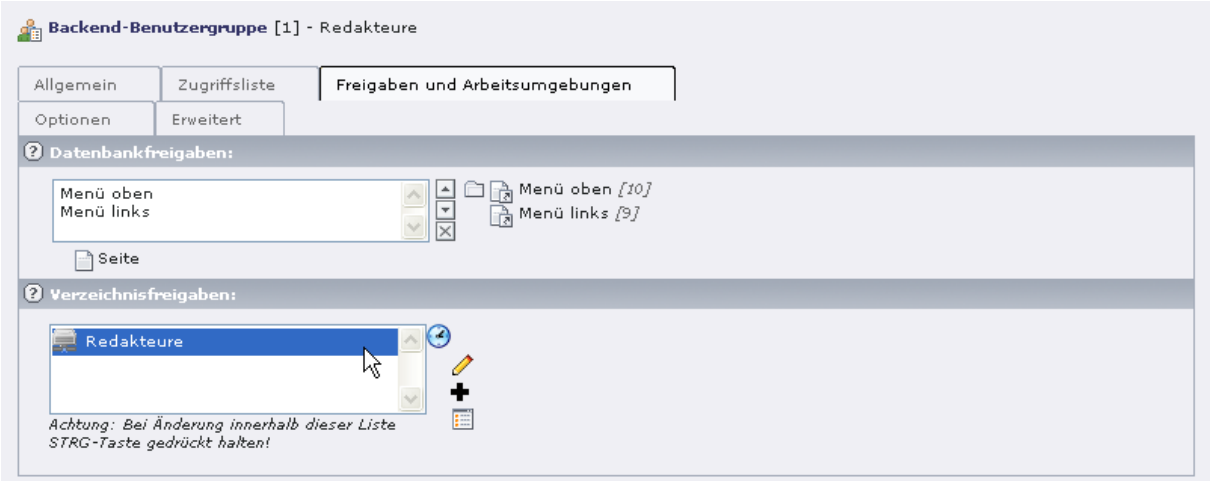
Im letzten Feld auf dem Reiter „Zugriffslisten“ können Sie den Benutzer auf die Bearbeitung einzelner Sprachen beschränken.

Speichern Sie den Datensatz und somit Ihre Einstellungen unbedingt, damit diese nicht aus Versehen verloren gehen und Sie alles noch einmal eingeben müssen. Nach dem Speichern wechseln wir auf den Reiter „Freigaben und Arbeitsumgebungen“. Im Feld Datenbankfreigaben legen wir den Einstiegspunkt im Seitenbaum fest. Wählen Sie die Seiten bzw. diejenigen Seiten aus, die der Redakteur bearbeiten soll. Sinnvoll ist:



Speichern Sie Ihre Eingaben erneut, um sicherzustellen, dass keine Angaben verloren gehen. Anschließend klicken wir unter „Verzeichnisfreigaben“ auf das Plus Icon, um eine neue Verzeichnisfreigabe zu erstellen. Dabei handelt es um die Freigabe eines Verzeichnisses unterhalb von fileadmin/, in dem der Redakteur arbeiten, also Dateien uploaden und auswählen darf:

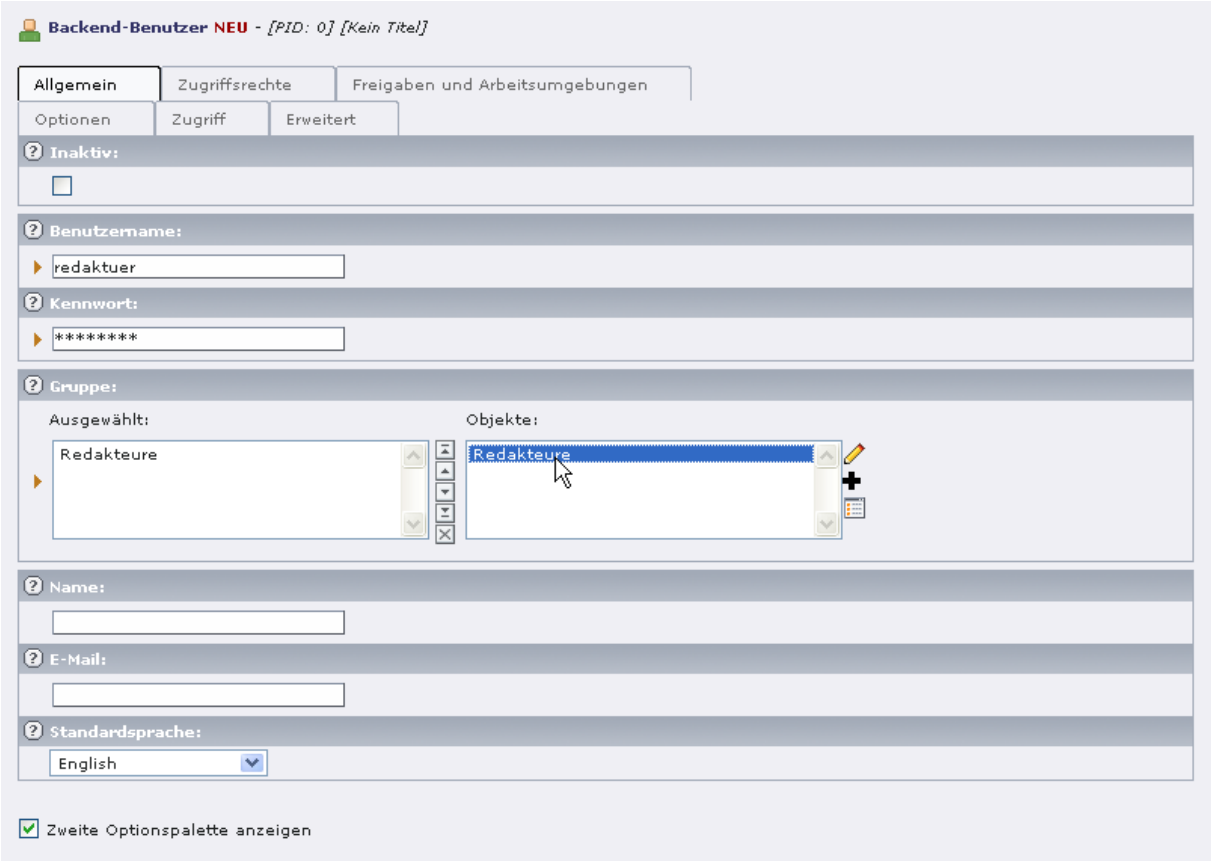
Wir geben einen Bezeichner „Redakteure“ und ein Zielverzeichnis „user_upload/“ an (der Ordner user_upload/ existiert in einer Standard-TYPO3-Installation bereits) und setzen die Pfadangabe relativ zum Ordner fileadmin/. Wir klicken auf die Schaltfläche „Speichern und Schließen“, um in die Benutzergruppen-Bearbeitungsmaske zurückzukehren.



Klicken Sie „Speichern und Schließen“, wir haben alle Angaben die zum Erstellen der Benutzergruppe benötigt werden, getätigt und werden im folgenden Kapitel einen Benutzer anlegen und der Gruppe hinzufügen.

7.9 Benutzer anlegen (Redakteur)

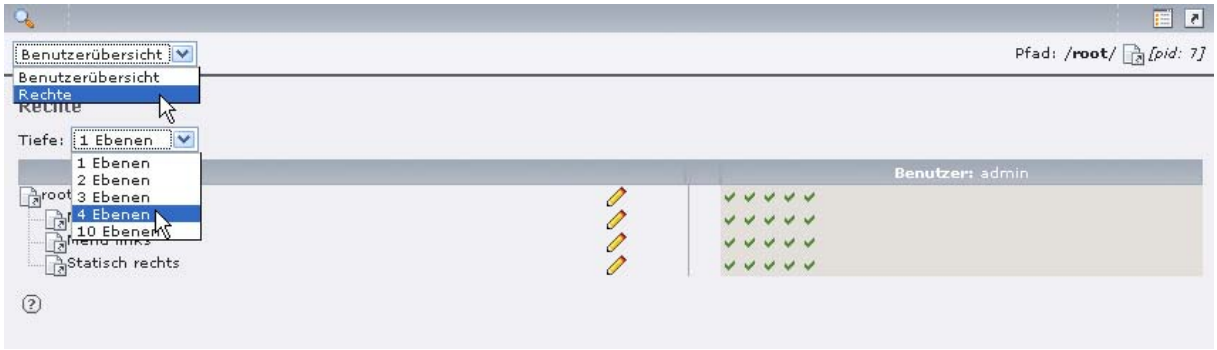
Wir legen einen neuen Datensatz vom Typ „Backend-Benutzer“ auf der Root-Ebene (Weltkugel) an. Vergeben Sie einen Benutzernamen und ein Passwort. Damit der Benutzer die Rechte der Benutzergruppe „Redakteure“ erhält, fügen wir im Feld „***Gruppe***“ die Benutzergruppe „***Redakteure***“ hinzu:



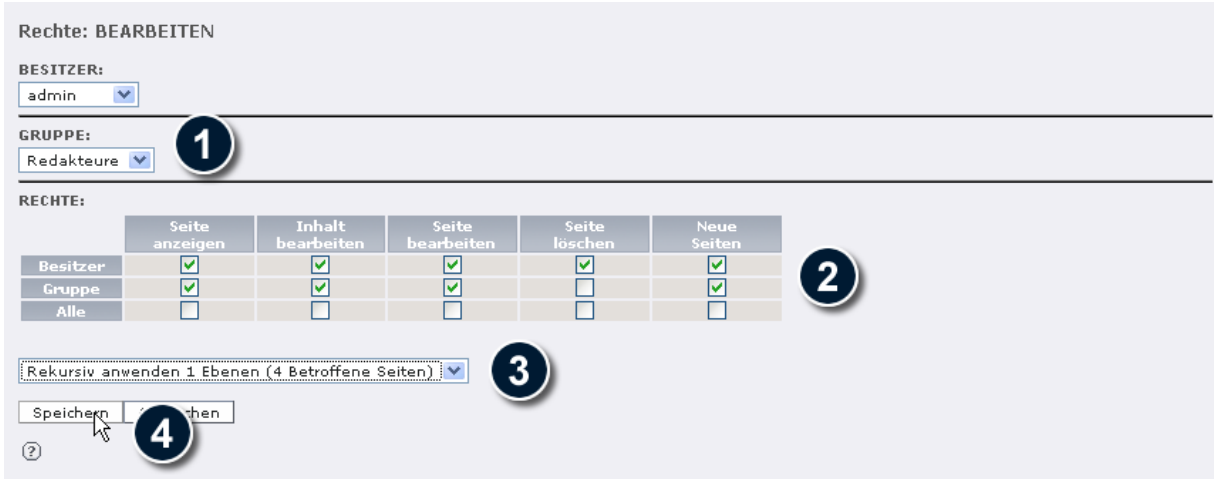
Wir speichern den Datensatz über die Schaltfläche „Speichern und schließen“.

7.10 Zugriffsrechte setzen

Nachdem wir nun eine sowohl eine Benutzergruppe als auch einen Benutzer angelegt haben, müssen wir allerdings noch für die entsprechenden Seiten Zugriffsrechte setzen (was leider oftmals vergessen wird). Wählen Sie aus der Auswahlbox den Eintrag „Rechte“ aus und setzten Sie die Auswahlbox „Tiefe“ auf „4 Ebenen“.



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 kann sich der neue Benutzer erfolgreich mit den gewünschten Möglichkeiten einloggen. Zum Testen gehen Sie in das Modul „***Liste***“ und auf die Root-Ebene (Weltkugel). Klicken Sie auf das Icon „grünes Männchen“ vor dem Backend-Benutzer „***Redakteur***“ und wählen Sie aus dem Pop-Up-Menü „***Switch to user***“ aus.

Wir werden nun automatisch als Benutzer „Redakteur“ angemeldet, erhalten ein eingeschränktes Backend mit den in der Benutzergruppe vergebenen Rechten. Über die Schaltfläche „Verlassen“ rechts oben können wir in unseren Admin-Bereich zurückkehren.

7.11 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, dass eine Statistik aussagt, dass 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, dass es sich direkt in die TYPO3-Umgebung integrieren lässt.

Damit man in den Genuss solcher Statistiken kommt, ist es erforderlich, mit TypoScript einige Werte zu setzen. Wir gehen davon aus, dass AWStats bereits installiert und betriebsbereit ist (also 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 Root-Template, wird um folgende Einträge ergänzt:

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

Die Bezeichnung stat_apache bzw. stat_apache_logfile mag ggf. irreführend sein. Es sei hier erwähnt, dass diese Datei im Format einer Apache-Logdatei erstellt wird, dies aber keineswegs die Apache-Logdateien ersetzt. Diese werden nach wie vor geschrieben und liegen 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, dass 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 zu aktualisieren, ist jedoch eine Leichtigkeit, indem man einfach auf den Link „Jetzt aktualisieren“ klickt.

Aber AWStats hat auch einige Nachteile: Es wird einem vorgetäuscht, dass 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 den besseren Kompromiss dar.

Ebenfalls sei erwähnt, dass 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, dass nicht einmal Grafiken mit berechnet werden. Angaben wie z.B. das Transfervolumen sind somit unbrauchbar.

8. TypoScript Kurzreferenz

8.1 Datentypen

Die Werte, die in TypoScript übergeben werden, sind von einem bestimmten Format.

8.1.1 Datentypen Übersicht

Die folgende Tabelle gibt Ihnen eine Übersicht über die Formate

Ist beispielsweise ein Typ als “<tag>“ angegeben, so muss HTML-Code angegeben werden. Ist der Typ beispielsweise „resource“, so muss eine Referenz zu einer Datei angegeben werden.

Eine Übersicht der meistgenutzten Typen finden Sie in der folgenden Tabelle:

Datentyp	Beispiel	Beschreibung	Default
<tag>	<BODY bgcolor=“red“>		
align	Right	right/left/center	left
VHalign	Hori.align = right and Vert.align = center: r , c	r/c/l , t/c/b Horizontal (right, center, left) , Vertical align (top/center/bottom)	
resource	Aus dem Resourcen Feld: toplogo*.gif Referenz auf das Dateisystem: fileadmin/picture.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	Die Datei ist eine imgResource:
	file = toplogo*.gif
	file.width = 200
	GIFBUILDER:
	file = GIFBUILDER
	file {
	... (GIFBUILDER-properties here)
	}
		1) Es handelt sich um eine "resource" (siehe Oben) zusätzlich sind imgResource-Eigenschaften vorhanden (siehe Beispiel und nachfolgende Objekt Beschreibung)
		imgResource erwartet als Datei eine gültige Grafik. Welche Dateitypen erlaubt sind, wird in [TYPO3_CONF_VARS[„GFX“] [„imagefile_ext“] festgelegt (Installtool bzw. localconf.php).
		
		2) GIFBUILDER-object	

HTML-code	<B>Test in Bold</B>	Reiner HTML Code	
target	_top _blank content	HTML-Target, der in <A>-Tags Verwendung findet. Wird überwiegend bei Frames benötigt. Der Datentyp target verdeutlicht, das ein HTMLTarget	

		erwartet wird. Als Wert wird jede beliebige Zeichenkette akzeptiert.	
imageExtension	jpg web(gif or jpg ..)	imageExtension kann jeder Typ sein, der in der globalen Variable $TYPO3_CONF_VARS["GFX"]["imagefile_ext"] (localconf.php) definiert wurde. Standard ist pdf,gif,jpg,jpeg,tif,bmp,ai,pcx,tga,png Der Wert „web“ gibt an,d as es sich um ein web Format (jpg oder gif) handelt.	
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.	
str/string/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(HTML-color) #ffeecc (HTML-color) 255,0,255(RGB-integers)	Wie HTML-color, jedoch ist zusätzlich die Angabe von RGB-Werten möglich.	
page_id	34 this	Angabe einer Seiten-ID. page-id Erwartet eine positive Zahl oder aber die Angabe von „this“ als Verweis auf die aktuelle Seite.	
pixels	345	Pixel Distanz	
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.	
linkWrap	Einen Link zur Root Seite erzeugen: <A HREF="?id={0}"> | </A>	Kann genutzt werden, um einen Link auf eine Seite zu erzeugen, z.B. um ein Bild zu verlinken. Als Zahl muss die ID einer Seit angegeben werden.	
case	upper	upper/lower Gibt an, ob eine Zeichenkette komplett groß- oder kleingeschrieben werden soll.	
space	5 | 5	"before | after" Gibt bei Inhalten den Abstand vor | nach dem Element an	
date-conf	d-m-y(dd-mm-yy format)	parallel der PHP-Funktion date()	

strftime-conf		parallel der PHP-Funktion strftime()	
UNIX-time	Seconds to 07/04 2000 23:58: 955144722	Sekunden seit dem 01.01.1970	
path	fileadmin/stuff/	Pfadangabe relativ zum Verzeichnis, von dem aus wir arbeiten	
getText		Erläuterungen siehe Abschnitt „data (getText)“	
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 rekursiv (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]	

8.1.2 data (getText)

[.data = ...]	
Mit der Funktion *data* und dem Datentyp *getText* können diverse Werte ausgelesen werden.	
field : [Feldname]	Mit *field* kann ein beliebiges Feld aus der Tabelle pages der aktuellen Seite ausgelesen werden. Identisch zur stdWrap- Funktion field. Beispiel: 10 = TEXT 10.data = field : header 20 = TEXT 20.field = header
getenv : HTTP_REFERER	Mit getenv 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 globalen 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 sind (z.B. im Backend sichtbar).

	Ebenfalls werden Datensätze, die als deleted gekennzeichnet sind, nicht ausgegeben.
LLL:EXT:tt_news/locallang.xml:pi_title	Mit LLL können Sprach Labels ausgelesen werden, z.B. wie in dem dargestellten Beispiel aus einer Extension.
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	

8.1.3 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

8.2 Objekte und Eigenschaften

8.2.1 Berechnungen (+calc):

Mit *+calc* können Berechnungen durchgeführt werden. Sobald als Datentyp *+calc* genutzt werden kann, können zusätzlich zu Zahlen auch Operatoren wie +, -, * etc. verwendet werden.

Beispiel (GIFBUILDER Grafik berechnen):

```
XY = [10.h]+10,[10.w]*3
```

8.2.2 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.

8.3 Conditions

8.3.1 Browser

[browser = ...]	
Microsoft Internet Explorer	msie
Netscape Communicator	netscape
Lynx	lynx
Opera	opera
Mozilla	Mozilla
PHP	fopen php
AvantGo	avantgo
Adobe	Acrobat WebCapture acrobat
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]	

8.3.2 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]

8.3.3 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]	

8.3.4 Devices

[device = ...]	
HandHeld/PDAs	pda
WAP-Handies	wap
Grabbers	grabber
Indexing robots	robot

8.3.5 Sprache

[language = ...]
Der Wert muß exakt mit dem Wert übereinstimmen, der in getenv(“http_ACCEPT_LANGUAGE“) in PHP gespeichert ist.

8.3.6 IP-Adressen

[IP =...]
Beispiele: [IP = 123.*.*.*] [IP = [192.168.*.*] [62.153.151.126]

8.3.7 Stunde

[hour = ...]
Beispiele: [hour = > 17]

8.3.8 Minute

[minute = ...]
Mögliche Werte 1..59 wie “Stunde”.

8.3.9 Wochentag

[dayofweek = ...]
Mögliche Werte 0..6 Sonntag = 0 bis Samstag = 6 Beispiele: [dayofweek =0]

8.3.10 Tag des Monats

[dayofmonth = ...]

Mögliche Werte 1..31 Beispiel: [dayofmonth = > 25]

8.3.11 Monat

[month = …]
Mögliche Werte 1..12 Beispiel: [month = 12]

8.3.12 Benutzergruppe (FE)

[usergroup = …]
Bei usergroup wird die uid der gewünschten FE-Benutzergruppe angegeben Beispiel: [usergroup = 16]

8.3.13 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 = *]

8.3.14 treeLevel

[treeLevel = …]
Prüft auf die aktuelle Ebene innerhalb des Baumes. Insb. für Manipulationen bei der Template-Vererbung sinnvoll einsetzbar. Beispiel: [treeLevel = 3]

8.3.15 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]

8.3.16 PIDupinRootline

[PIDupinRootline = …]
Wie „PIDInRootline“. Die aktuelle Seite wird hier jedoch ausgeschlossen.

8.3.17 GlobalVar/GlobalString

[globalVar = ...] **[globalString = ...]**	
GP:varname	GET/POST Variablen
ENV:varname	Umgebungsvariablen
IENV:varname	Besser als ENV, da plattformübergreifend
LIT:litwert	Prüfen geben einen literalen Wert
TSFE:varname	Variablen aus TSFE – TypoScript Frontend Engine
Prüft auf globale Variablen (GET, POST, Umgebungs-Variablen) IENV, ENV, GP, TSFE, LIT Beispiele: [globalString = HTTP_HOST=www.typo3server.com] [globalString = ENV:REMOTE_ADDR=192.168.*.*] [globalVar = TSFE:id > 10]	

8.3.18 compatVersion

[compatVersion = branch-version

Beispiel:
#Gilt nur für TYPO3 Version 4.2
[compatVersion = 4.2]

8.3.19 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()]

8.4 Funktionen

8.4.1 stdWrap - Daten auslesen (get data)

Eigenschaft	Datentyp	Beschreibung
setContentToCurrent	boolean	Kann mit einem Speicher verglichen werden. *current* wird mit dem Inhalt des aufrufenden Elementes gefüllt. Bei einem anderen Element kann so mit current = 1 dieser Inhalte erneut erzeugt werden.
setCurrent	string /stdWrap	Setzt den Wert für current.
lang	string	Die Eigenschaft wird genutzt, um alternative Sprachinhalte zu setzten. Sprachkürzel können sein de, fr, en, … Die über die globale Eigenschaft *config.language* spezifizierte Sprache wird über die Eigenschaft *lang* gesetzt: **Beispiel**: config.language = de page.10 = TEXT page.10.value = I am a Berliner! page.10.lang.de = Ich bin ein Berliner! Ausgabe ist “Ich bin...” anstelle von “I am...”

data	getText	Die Funktion hat einen großen Umfang, nähere Informationen daher im Abschnitt „data (getText)“
field	fieldname/string	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.
current	boolean	Setzt den Inhalt auf den aktuellen Wert
cObject	cObject	Der Inhalt eines anderen cObjektes (z.B. TEXT, IMAGE etc.) wird ausgegeben.
numRows	numRows	Liefert die Anzahl der Datenbankeinträge zurück.
filelist	dir /stdWrap	Liest ein Verzeichnis und gibt eine Liste der enthaltenen Dateien wieder.
preUserFunc		Ruft eine PHP Funktion oder Methode einer Klasse auf. Im ersten Parameter wird der Inhalt im zweiten alle übrigen Parameter übermittelt.

8.4.2 stdWrap - Daten überschreiben/Bedingungen (override/conditions)

Eigenschaft	Datentyp	Beschreibung
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).
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

8.4.3 stdWrap-Daten verarbeiten (parse Data)

Eigenschaft	Datentyp	Beschreibung
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.
removeBadHTML	boolean	Entfernt, wenn gesetzt, gefährlichen HTML-Code, der XSS ermöglicht.
stripHtml	boolean	Entfernt alle HTML-Tags aus dem Inhalt htmlSpecialChars boolean Der Inhalt wird durch die PHP-Funktion htmlspecialchars() geparst. Die Auswirkungen sind, dass HTML-Code nicht ausgeführt, sondern direkt angezeigt wird.
crop	string	Schneidet den Inhalt nach einer bestimmten Anzahl von Zeichen ab und fügt beliebige Zeichen an. **Syntax:** Anzahl | End Text | Wortbruch Ist die Anzahl der Zeichen positiv, wird vom Anfang des Strings gerechnet, bei einer negativen Zahl vom Ende aus. Der „End Text“ wird angehängt und mit „Wortbruch = 1“ kann angegeben werden, ob ganze Wörter erhalten werden sollen. **Beispiel:** Dieses Beispiel liefert als Ergebnis „Dies ist nur...“ 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 | ... | 1
rawUrlEncode	boolean	Auf den Inhalt wird die PHP Funktion rawUrlEncode() angewendet: Gibt einen String zurück, in dem alle nicht-alphanumerischen Zeichen außer -_. durch ein Prozentzeichen (%) gefolgt von zwei Hex-Werten ersetzt wurden.
htmlSpecialChars	boolean	Auf den Inhalt wird die PHP Funktion htmlSpecialChars

		() angewendet: Wandelt Sonderzeichen in HTML-Codes um.
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 durch dem angegebenen Wert ersetzt. **Beispiel**: doubleBrTag = <p>
addParams		Mit der Funktion können Attribute in HTML Tags eingefügt werden. **Beispiel**: 10 = HTML 10.value = < table > 10.addParams.align = center
typolink		Nähere Informationen im Abschnitt „typolink“ (Seite 183).

8.4.4 stdWrap - Datums- und Zeitfunktionen

Eigenschaft	Datentyp	Beschreibung
date	date-conf	Wandelt einen UNIX Timestamp in ein lesbares Datumsformat um. **Beispiel**: seite.10 = TEXT seite.10.value.field = tstamp seite.10.value.date = d-m-y
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 zurückgelieferten 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.

8.4.5 stdWrap - EditPanel

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

8.4.6 stdWrap - Debugging

Eigenschaft	Datentyp	Beschreibung
debug	boolean	Ähnlich wie htmlSpecialChars: Liefert das Ergebnis in HTML-Darstellung zurück. HTML-Code wird somit nicht

		vom Browser ausgeführt.
debugFunc	Boolean/Integer	Ähnliche Ausgabe wie bei der Funktion debug, allerdings muss die eigene IP Adresse in der „devIPmask“ gesetzt sein. Mit debugFunc = 2 können einzelne Werte in einer Tabelle überprüft werden.
debugData	boolean	Liefert den Inhalt von $cObj->data direkt an den Browser aus, allerdings muss die eigene IP Adresse in der „devIPmask“ gesetzt sein.

8.4.7 imgResource

Eigenschaft	Datentyp	Beschreibung
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

8.4.8 imageLinkWrap

Eigenschaft	Datentyp	Beschreibung
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)

typolink	->typolink	Nähere Informationen im Abschnitt „typolink“ (Seite 183).
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
JSwindowexpand	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.

8.4.9 numRows

Eigenschaft	Datentyp	Beschreibung
table	string	Angabe des Tabellennamens
select	-> select	Nähere Informationen im Abschnitt “select”.

8.4.10 select

Eigenschaft	Datentyp	Beschreibung
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’

8.4.11 split

Eigenschaft	Datentyp	Beschreibung
token	string/stdWrap	Trennzeichen das für den split verwendet wird.
max	int/stdWrap	Maximale Anzahl an Aufteilungen
min	int/stdWrap	Minimale Anzahl an Aufteilungen
1,2,3,4	-> CARRAY/stdWrap	Das Objekt, das den Wert bearbeiten soll. **Beispiel:** 1.current = 1

		1.wrap = |
cObjNum	cObjNum + optionSplit	Ein Pointer über den man die Element aus dem Array ("1,2,3,4"), anspricht.
wrap	wrap + optionSplit	Gibt einen Wrap für jeden einzelnen Eintrag an.

8.4.12 if

Eigenschaft	Datentyp	Beschreibung
Die „if-Funktion“ liefert „wahr“ zurück, wenn alle(!) verwendeten 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.

8.4.13 typolink

Eigenschaft	Datentyp	Beschreibung
extTarget	target/stdWrap	Gibt das target/ziel bei externen Links an.
target	target /stdWrap	Gibt das target/ziel bei internen Links an.
no_cache	boolean /stdWrap	Fügt "&no_cache=1"-den Link hinzu.
useCacheHash	boolean	Wenn gesetzt wird für jede Kombination aus Parametern Cache-Eintrag geschrieben, indem ein cHash-Wert an die URL gehangen wird. Der Wert [SYS][encryptionKey] wird in den hash verarbeitet um diesen eindeutig und nicht berechenbar zu machen.
additionalParams	string /stdWrap	Die angegeben Parameter werden an das Ende der URL angehängt. Der String muss also komplett mit einem vorangestellten „&“ angegeben werden: **Beispiel**: '&print=1' **Hinweis**: Die Funktion kann nur für interne Links genutzt werden.
addQueryString	boolean	Fügt den QUERY_STRING am Ende eines Links hinzu.

		Hinweis: Um doppelte Parameter zu verhindern, sollte config.uniqueLinkVars gesetzt werden.
wrap	wrap	Wrappt den Link
ATagBeforeWrap	boolean	Wenn gesetzt, wird zuerst der wrap ausgeführt und anschließend mit dem A Tag umschlossen.
parameter	string /stdWrap	Es werden die Daten angegeben, die zur Linkerzeugung verwendet werden: **Interne Links:** Wir eine Integer-Zahl angegeben, so wird dies als Seiten-ID gewertet. Wird nach einem Komma eine weitere Zahl angegeben, so wird diese als Pagetype ausgewertet. **Externe Links:** Werden mit einem http:// versehen, E-Mail-Adressen können angegeben werden.
title	string /stdWrap	Setzt das title Tag im Link.
section	string /stdWrap	Setzt ein # um Sprungziele zu erzeugen.
ATagParams	<A>-params /stdWrap	Es können zusätzliche Parameter für den A Tag gesetzt werden.

8.4.14 encapsLines

Eigenschaft	Datentyp	Beschreibung
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 Großbuchstaben angegeben werden!

8.4.15 parseFunc

Eigenschaft	Datentyp	Beschreibung
short	Strings (Array)	Hier können Abkürzungen vereinbart werden. **Beispiel**: .short.t3 = Typo3 .short.backToHome = <a href=”index.php”> Alle Vorkommnisse von “t3” werden in „Typo3“ umgewandelt. Das zweite 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	Liste von Strings (Kommasepariert)	In „denyTags“ können HTML-Tags verboten werden. Beispiel: .denyTags = font, span
allowTags	Liste von Strings (Kommasepariert)	Hier werden erlaubte Tags aufgeführt. Wird ein Tag in „allowTags“ gefunden, wird „denyTags“ ignoriert. **Beispiel:** .allowTags = b,i,a,img .denyTags = * Folgende Tags werden so erlaubt <B>, <I>, <A> und <IMG>.

8.5 Setup

8.5.1 CONFIG

Eigenschaft	Datentyp	Beschreibung
linkVars	stringliste	Variablen aus HTTP_GET_VARS, die immer an eine erzeugte URL angefügt werden sollen. Beispiel: config.linkVars = session Dies fügt an einen Link den Parameter „session“ mit einem übertragenen Wert. Hinweis: Einige Module/Erweiterungen erzeugen direkte Links, hierdurch kann diese Eigenschaft ggf. unbrauchbar werden.
uniqueLinkVars	boolean	Wenn gesetzt, wird unterbunden, dass Parameter doppelt in der URL enthalten sind.
renderCharset	string	Gibt den Zeichensatz an, der für das Redner der Seiteninhalte verwendet wird.
metaCharset	string	Gibt die Zeichencodierung an, die für die Ausgabe verwendet wird. Die Angabe wird im Header im <meta> Tag erzeugt.
doctype	string	Setzt den Doctype. Mögliche Werte: none – kein Doctype xhtml_11 – XHTML 1.1 xhtml_20 – XHTML 2.0 xhtml_frames – XHTML Frameset xhtml_trans – XHTML Transitional xhtml_strict – XHTML Strict
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
intTarget	target	Setzt das Ziel/Target für interne Links
extTarget	target	Setzt das Ziel/Target für externe Links
spamProtectEmailAddresses	"ascii" / integer	Wenn diese Eigenschaft gesetzt wird, werden alle E-Mail-Adressen verschlüsselt, so dass E-Mail-Robots diese E-Mail-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 aktiviert, 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-Benutzern 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 DBTabelle – sysstat Hinweis: Die Aktivierung kann große Tabellen mit vielen Einträgen erzeugen! 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.
language	string	Setzt die Sprache der Seitenausgabe (z.B. de, en, fr, …)
sys_language_uid	integer	Setzt die jeweilige Sprache, indem der Wert auf die UID des Datensatzes „Webseitensprache“ eingestellt wird.
notification_email_encoding	string	Hier kann für versendete E-Mails (plaintext) der

		jeweilige encode-Modus angegeben werden. Mögliche Werte sind: base64, quoted-printable, 8bit
notification_email_urlmode	string	Bei versendeten E-Mails können Links lang werden. Um einen Zeilenumbruch zu verhindern, 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.

8.5.2 PAGE

Eigenschaft	Datentyp	Beschreibung
typeNum	Integer	Angabe der typeNum, die das gewünschte „PAGE“-Objekt auswählt. Der default-Wert für das PAGE-Objekt ist 0, sofern keine typeNum angegeben wird.
1,2,3,10,20,30	cObject	definiert die Reihenfolge der Ausgabe der Inhaltselemente.
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”>
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)	Hier wird ein Verweis zu einer .ico-Datei angegeben. das Icon kann in manchen Browsern dargestellt werden.
headerData	string	Fügt beliebigen Code (z.B. JavaScript-Code etc.) in den Header-Bereich ein.
config	-> CONFIG	Nähere Informationen ab Abschnitt „CONFIG“
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:
		includeCSS {
		file1 = fileadmin/mystylesheet1.css
		file2 = stylesheet_templateressource*.css
		file2.title = High contrast
		file2.media = print
		}

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.

8.5.3 FE_DATA/FE_TABLE

Eigenschaft	Datentyp	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, das 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.

8.5.4 FRAME

Eigenschaft	Datentyp	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

8.5.5 FRAMESET

Eigenschaft	Datentyp	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"

Beispiel

```
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"
}
```

8.5.6 META

Eigenschaft	Datentyp	Beschreibung
Array...	string /stdWrap	Mit dem META-Objekt können seitenübergreifende Meta Tags gesetzt werden. Es können Keywords verwendet werden wie (DESCRIPTION, AUTHOR,KEYWORDS) Beispiel: meta.DESCRIPTION = Beschreibungstext ergibt: <meta name=“DESCRIPTION“ content“Beschreibungstext“>

8.6 Objekt-Referenz

8.6.1 TEXT

Eigenschaft	Datentyp	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

8.6.2 COBJ_ARRAY (COA, COA_INT)

Eigenschaft	Datentyp	Beschreibung
Wenn anstelle von COA das COA_INT Objekt genutzt wird, werden die zusammengefassten Objekte nicht gecachet.		
1,2,3,4,10,20	cObject	Mit COA kann aufgesplittert 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	wrappt das komplette COA
stdWrap	stdWrap	stdWrap-Funktion können genutzt werden

8.6.3 FILE

Eigenschaft	Datentyp	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	

8.6.4 IMAGE

Eigenschaft	Datentyp	Beschreibung
file	imgResource	
params	string	Angabe von Parameter für den img-Tag
border	integer	Gibt die Breite des Rahmens an.
alttext titleText	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	
imageLinkWrap	boolean/ ->imageLinkWrap	Erzeugt einen Link um das <img> Tag, nur aktiv, wenn linkwrap nicht genutzt wird. Siet TYPO3 version 4 muss

		expliziet die Option imageLinkWrap = 1 gestzt werden um die Funktin nutzen zu können
linkWrap	linkWrap	Wrappt den <img> Tag mit einem Link. Besser ist stdWrap/typolink oder imageLinkWrap zu nutzen.

8.6.5 IMG_RESOURCE

Eigenschaft	Datentyp	Beschreibung
Erzeug ähnlich wie IMAGE ein Bild, jedoch wird nur der Pfad zurückgeliefert.		
file	imgResource	
stdWrap	stdWrap	

8.6.6 CONTENT

Eigenschaft	Datentyp	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“
wrap	wrap	
stdWrap	stdWrap	

8.6.7 RECORDS

Eigenschaft	Datentyp	Beschreibung
Über das RECORDS Objekt können einzelne Einträge aus Tabellen der Datenbank gelesen werden.		
source	records-list /stdWrap	Liste der Datensatz IDs, optional Angabe der Tabellennamen **Beispiel**: tt_content_34, 45, tt_links_56.
tables	list of tables	Liste der Tabellen, von denen die Einträge geholt werden sollen

8.6.8 HMENU

Eigenschaft	Datentyp	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 mögliche Werte sind: "directory"/"list"/"updated"/"browse"/"rootline"/"keywords"/“language”	Mit special können die Menüelemente selektiert werden. Diese Eigenschaft ist nicht zwingend erforderlich, für manche Realisierungswünsche ist sie jedoch sehr nützlich. **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 (kommaseparierte 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	

8.6.9 CASE

Eigenschaft	Datentyp	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 TEXTObjekt 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.

8.6.10 FORM

Eigenschaft	Datentyp	Beschreibung
layout	string	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	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 Pflichtfelder 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	E-Mail-Adresse des Empfängers, wenn bei der Konfiguration formtype_mail=submit angegeben wurde.
data	string /stdWrap	Hier stehen die einzelnen Formularfelder. Wird oft in Kombination mit „field“ verwendet, z.B. data.field = Datenbankfeld.

dataArray	[array of form elements]	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

8.6.11 USER/USER_INT

Der Unterschied zwischen USER und USER_INT liegt insbesondere an der Handhabung des Caches. Das Resultat aus USER_INT wird nicht gecachet. 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	Datentyp	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

8.6.12 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.

Eigenschaft	Datentyp	Beschreibung
file	Resource / Datei	Datei, die inkludiert wird. Die Datei muss validen PHP-code enthalten! Der Code wir mittels "include()" eingefügt. 1) Der gesamte Inhalt/ die Ausgabe muss in $content geladen werden, es muss nicht mittels echo ausgegeben werden. 2) Das Caching sollte in der Entwicklungsphase deaktiviert werden über $GLOBALS["TSFE"]->set_no_cache().

8.6.13 TEMPLATE

Eigenschaft	Datentyp	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

subparts	Array... mehrerer cObject	Subparts werden mit dieser Eigenschaft mit Inhalt gefüllt bzw. der darin enthaltenen ersetzt.
		Ein Subpart wird definiert auf zwei Markern. Die Marker müssen mit „###“ umschlossen werden und in HTML-Kommentaren stehen
		Beispiel:
		subparts {
		HELLO = TEXT
		HELLO.value = Ein TEXT
		}
		Im Template:
		<!-- start of subpart: ###HELLO### -->
		Hier steht der HTML Text der überschrieben wird.
		<!-- end ###HELLO### -->
		Hinweis:
		Die Platzhalterbezeichner sind casesensitive!

marks.[Platzhalter]	Array... mehrerer 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 casesensitive!
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.

8.6.14 EDITPANEL

Eigenschaft	Datentyp	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 (Kommasepariert)	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.

8.7 GIFBUILDER

Eigenschaft	Datentyp	Beschreibung
1,2,3,4...	GifBuilderObj	Gibt die Ebenen an, auf die Gifbuilder Objekt wie Text, Shadow usw. geladen werden.
XY	x, y (Kalkulation möglich)	Mit XY werden die gesamten Dimensionen der zu erzeugenden Grafik angegeben.
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	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	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
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

8.8 Menue Objekte

8.8.1 Allgemeine Eigenschaften

Eigenschaft	Datentyp	Beschreibung
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. Hinweis: Hat Vorrang vor HMENU.minItems
maxItems	integer	Die maximale Anzahl von Menüeinträgen. Falls mehr Menüeinträge vorhanden sind, werden diese bei der Menüerzeugung ignoriert. Hinweis: Hat Vorrang vor HMENU.maxItems
begin	integer	Der erste Eintrag im Menü. **Beispiel**: Um die ersten zwei Elemente eines Menüs nicht anzuzeigen, kann man hier begin = 3 angeben. Hinweis: Hat Vorrang vor HMENU.begin
addParams	string	Mit addParamt können zusätzliche Parameter den Links des Menüs hinzugefügt werden.
alternativeSortingField		In der Regel wird die Sortierung der Menüpunkte durch TYPO3- die Tabelle *pages* vorgegeben. Die Sortierung kann auch durch ein anderes Feld erfolgen, z.B. nach dem Datum, dem Titel etc.

8.8.2 Allgemeine Menüzustände

Eigenschaft	Datentyp	Beschreibung
NO	boolean/(config)	Standard-Beschreibung des Menüs
IFSUB	boolean/(config)	Beschreibung der Menüeinträge, die Unterseiten besitzen
IFSUBRO	boolean/(config)	Beschreibung der Menüeinträge, die Unterseiten besitzen, wenn ein Roll Over der Maus erfolgt – RollOver Effekt
ACT	boolean/(config)	Beschreibung der Menüeinträge, die sich in der Rootline befinden.
ACTRO	boolean/(config)	Beschreibung der Menüeinträge, die sich in der Rootline befinden, wenn ein Roll Over der Maus erfolgt

		– RollOver Effekt.
ACTIFSUB	boolean/(config)	Beschreibung der Menüeinträge, die sich in der Rootline befinden und die Unterseiten besitzen
ACTIFSUBRO	boolean/(config)	Beschreibung der Menüeinträge, die sich in der Rootline befinden und Unterseiten besitzen, wenn ein Roll Over der Maus erfolgt – RollOver Effekt.
CUR	boolean/(config)	Beschreibung der Menüeinträge der aktuellen Seite.
CURRO	boolean/(config)	Beschreibung der Menüeinträge der aktuellen Seite, wenn ein Roll Over der Maus erfolgt – RollOver Effekt.
CURIFSUB	boolean/(config)	Beschreibung der Menüeinträge der aktuellen Seite, die Unterseiten besitzen.
CURIFSUBRO	boolean/(config)	Beschreibung der Menüeinträge, der aktuellen Seite, die Unterseiten besitzen, wenn ein Roll Over der Maus erfolgt – RollOver Effekt.
USR	boolean/(config)	Beschreibung von Menüeintragen, auf die nur angemeldete Benutzer Zugriff haben.
USRRO	boolean/(config)	Beschreibung von Menüeintragen, auf die nur angemeldete Benutzer Zugriff haben, wenn ein Roll Over der Maus erfolgt – RollOver Effekt.
SPC	boolean/(config)	Beschreibung von Platzhaltern – TYPO3 Space

8.8.3 GMENU

Eigenschaft	Datentyp	Beschreibung
expAll	boolean	Wenn gesetzt, werden alle Untermenüs aufgeklappt.
collapse	boolean	Wenn gesetzt, werden aktive Menüelemente, welche die nächste Eben ausgeklappt haben, bei erneutem Klick geschlossen.
noBlur	boolean	Entfernt das gestrichelte Kästchen um die Links im IE.
target	target	Setzt das target der Menülinks.
stdWrap	->stdWrap	
min	x,y+calc	Skaliert das Menü in die angegebene minimale Größe.
max	x,y+calc	Skaliert das Menü in die angegebene maximale Größe.
useLargestItemX	boolean	Wenn gesetzt, wird die Breite des Menüs auf die des breitesten Menüeintrages gesetzt.
useLargestItemY	boolean	Wenn gesetzt, wird die Höhe des Menüs auf die des höchsten Menüeintrages gesetzt.

8.8.4 GMENUITEM

Eigenschaft	Datentyp	Beschreibung
noLink	boolean	Wenn gesetzt, wird kein Link gesetzt.
imgParams	params	Setzten von <img> Attributen
altTarget	string	Setzt ein alternatives Target
altImgResource	imgResouce	Angabe eines alternativen Bildes
ATagParams	string /stdWrap	Zusätzliche Attribute für den <a> Tag
ATagTitle	string /stdWrap	Setzt *title* Attribut in den <a> Tag
additionalParams	string /stdWrap	Zusätzliche Parameter für den Link

wrap	wrap	wrappt das Element
allwrap	string /stdWrap	wrappt das gesamte Element
allStdWrap	-> stdWrap	Stellt stdWrap Funktionen für das gesamte Element bereit.

8.8.5 TMENU

Eigenschaft	Datentyp	Beschreibung
expAll	boolean	Wenn gesetzt, werden alle Untermenüs aufgeklappt.
collapse	boolean	Wenn gesetzt, werden aktive Menüelemente, welche die nächste Eben ausgeklappt haben, bei erneutem Klick geschlossen.
noBlur	boolean	Entfernt das gestrichelte Kästchen um die Links im IE.
target	target	Setzt das target der Menülinks.
wrap	wrap	wrappt das Element
allwrap	string /stdWrap	wrappt das gesamte Element
allStdWrap	-> stdWrap	Stellt stdWrap Funktionen für das gesamte Element bereit.

8.8.6 TMENUITEM

Eigenschaft	Datentyp	Beschreibung
allWrap	string /stdWrap	wrappt das gesamte Element
wrapItemAndSub	string /stdWrap	wrappt den Menüeintrag und enthaltene Untermenüs
before	HTML /stdWrap	HTML Code der vor das Element geschrieben wird.
beforeImg	imgResource	Bild das vor das Element gesetzt wird <img> Tag
beforeImgTagParams	<img>-params	Attribute für das <img> Tag.
beforeImgLink	boolean	Wenn gesetzt, wird das *beforeImg* verlinkt
beforeROImg	imgResource	Setzt ein RollOver Bild für das *beforeImg*
beforeWrap	wrap	wrappt das *beforeImg*
linkWrap	wrap	wrappt den Link
stdWrap	-> stdWrap	
ATagBeforeWrap	boolean	Wenn gesetzt, wird der <a> Tag vor dem wrap geschrieben.
ATagParams	<A>-params /stdWrap	Zusätzliche Attribute für den <a> Tag
ATagTitle	string /stdWrap	Setzt *title* Attribut in den <a> Tag
additionalParams	string /stdWrap	Zusätzliche Parameter für den Link
doNotLinkIt	boolean	Eintrag wird nicht verlinkt

8.9 PageTSConfig : Seiten

TypoScript wird im TYPO3 Backend nicht nur für die Konfiguration des Frontends verwendet, sondern auch für Einstellungen des Backends. In den Seiteneigenschaften einer Seite befindet sich ein TSConfig Feld, indem Sie das PageTSconfig eintragen um Konfigurationen des Backend auszuführen.

8.9.1 mod.[modulname]

Über mod werden Backend Module konfiguriert. Der Modulname ist in der *conf.php* Datei des jeweiligen Moduls zu finden.

SHARED (gültig für alle Module)		
Eigenschaft	Typ	Beschreibung
SHARED.colPos_list	string Liste	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
web_layout (Web -> Seite)		
Eigenschaft	Typ	Beschreibung

web_layout.menu.function	[array].bolean	0 => QuickEdit
		1 => Columns
		2 => Languages
		3 => Seiten Information
		Beispiel:
		Deaktiviert alle Einträge bis auf "QuickEdit":
		mod.web_layout.menu.function {
		1 = 0
		2 = 0
		3 = 0
		}

web_layout.tt_content.fieldOrder	string Liste	Sortierreihenfolge im Backend für tt_content-Elemente. (kommaseparierte Liste).
web_layout.noCreateRecordsLink	boolean	Der Link „Neuen Datensatz anlegen“ wird nicht angezeigt.
web_layout.disableBigButtons	boolean	Deaktiviert die großen Schaltflächen in der Spaltenübersicht
web_layout.tt_content.colPos_list	string Liste	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
web_info (Web > Info)		
Eigenschaft	Typ	Beschreibung
web_info.menu.function	[array].boolean	Legt die Sichtbarkeiten von Optionen im Funktionsmenü fest. tx_cms_webinfo_page => Pagetree overview tx_belog_webinfo => Log tx_infopagetsconfig_webinfo => Page TSconfig tx_cms_webinfo_hits = > Hit Statistics tx_indexedsearch_modfunc1 =>Indexed search **Beispiel**:

		Deaktiviert den Eintrag „Hit Statistics“
		mod.web_info.menu.function {
		tx_cms_webinfo_hits = 0
		}

web_func (Web>Functions)		
Eigenschaft	Typ	Beschreibung
web_func.menu.function	[array].boolean	Legt die Sichtbarkeiten von Optionen im Funktionsmenü fest. tx_funcwizards_webfunc => Wizards

web_func.menu.wiz	[array].boolean	**Web>Functions>Wizards Untermenü**
		
		tx_wizardcrpages_webfunc_2 => Create multiple pages
		tx_wizardsortpages_webfunc_2=> Sort pages
		Beispiel:
		Deaktiviert den Eintrag "Mehrere Seiten erzeugen":
		mod.web_func.menu.wiz {
		tx_wizardcrpages_webfunc_2 = 0
		}

web_ts (Web -> Template)		
Eigenschaft	Typ	Beschreibung
web_ts.menu.function	[array].boolean	Legt die Sichtbarkeiten von Optionen im Funktionsmenü fest. tx_tstemplateceditor => Constant Editor tx_tstemplateinfo => Info/Modify tx_tstemplateobjbrowser => TypoScript Object Browser tx_tstemplateanalyzer =>Template Analyzer tx_tstemplatestyler_modfunc1 => CSS Styler
web_list (Web -> Liste)		
Eigenschaft	Typ	Beschreibung
web_list.alternateBgColors	boolean	Mit aktivierter Eigenschaft werden Hintergründe bei Elementen alternierend dargestellt
web_list.newWizards	boolean	Aktiviert bzw. deaktiviert den „Neuen-Datensatz-Anlegen“-Wizard
web_list. noCreateRecordsLink	boolean	Der Link „Neuen Datensatz anlegen“ wird nicht angezeigt.
web_list. hideTables	string Liste	Die in der Liste genannten Tabellen werden als Datensätze nicht angezeigt.

web_list. allowedNewTables	string Liste	Wenn diese Liste genutzt wird, können nur Datensätze über die Schaltfläche „Create New Record“ in diesen Tabellen eingefügt werden
		Beispiel:
		Nur Datensätze vom Typ *pages* und *tt_news* im “Neuen Datensatz” Maske angelegt werden
		mod.web_list {
		allowedNewTables = pages, tt_news
		}

8.9.2 TCEMAIN

Über TCEMAIN kann die TYPO3 Core Engine konfiguriert werden, diese ist für das Speichern von Datensätzen zuständig.

Eigenschaft	Typ	Beschreibung
clearCache_disable	boolean	Wenn gesetzt, wird der Cache nicht automatisch gelöscht nach dem Speichern.
clearCache_pageGrandParent	boolean	Wenn gesetzt, wird der Cache auch von Großelternseiten gelöscht.
clearCache_pageSiblingChildren	boolean	Wenn gesetzt, wird der Cache auch von Geschwisterseiten, also Seiten, die die gleiche Paret ID haben, gelöscht.
clearCacheCmd	string Liste | pages | all	Löscht den nach dem Speichern für die angegebene Seite. all bedeutet, dass der gesamte Cache gelöscht wird, pages, dass nur der Seitencache geleert wird.
permissions.groupid		Neu angelegte Seiten werden den hier angegebenen Gruppen ID zugeordnet.
permissions.userid		Neu angelegte Seiten werden dem User mit der hier angegebenen User ID zugeordnet.

permissions.user
permissions.group
permissions.everybody
	liste von strings	Setzen von Standard-Rechten für den Besitzer, die Gruppe und alle anderen Benutzer auf Seitenebene.
		Mögliche Werte: show,edit,delete,new,editcontent
		Standard-Rechte sind:
		"user" => "show,edit,delete,new,editcontent",
		“group" => "show,edit,new,editcontent",
		"everybody" => ""
		Beispiel:
		TCEMAIN.permissions {
		# Besitzer hat alle Rechte (default):
		user = show,edit,delete,new,editcontent
		# Die Gruppe bekommt jetzt alle Rechte
		# (normalerweise ist "delete" deaktiviert)
		group = show,edit,delete,new,editcontent
		# Alle anderen dürfen die Seite sehen
		everybody = show
		}

8.10 UserTSConfig : Benutzer

Das UserTSConfig kann im TSConfig Feld von Benutzern und Benutzergruppen eingetragen und so Konfigurationen durchgeführt werden. Konfigurationen auf Benutzer-Ebene (UserTSConfig) haben Vorrang vor der Konfiguration auf Seiten-Ebene (PageTSConifg).

8.10.1 admPanel

Über admPanel wird das Admin Panel im Frontend konfiguriert.

Eigenschaft	Typ	Beschreibung
Hinweis: Damit das Adminpanel angezeigt und aktiviert wird, muss ebenfalls in TypoScript im Template config.admPanel aktiviert werden.		
enable.[bereich]	boolean	Hier können verschiedene Bereiche des Adminpanels aktiviert werden. 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

8.10.2 options

Setzen von unterschiedlichen Optionen für den Benutzer.

Eigenschaft	Typ	Beschreibung
RTEkeyList	string Liste (kommasepariert)	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 Liste (kommasepariert)	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 Liste (kommasepariert)	Hier können die einzelnen Elemente aus den Klickmenüs (Element-Browser) deaktiviert werden. key kann folgende Wert annehmen; pageTree, pageList, folderTree, folderList, page und folder 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
additionalPreviewLanguages	string Liste	Liste von UIDs der Sprachen, die der Benutzer sehen kann.
alertPopups	string	Konfiguration der Java Script PopUp Meldungen. Konfiguration über Bit Maske: Bit 1 = Ändern des Seitentyps Bit 2 = Kopieren/Verschieben, Einfügen Bit 3 = Löschen Bit 4 = FE-Editing
createFoldersinEB	boolean	Ermöglicht das Anlegen von Ordnern über den Element Browser.
defaultFileUploads	integer	Anzahl der Dateien, die in der Dateiliste auf den Server geladen werden können.

8.10.3 setup

Eigenschaft	Typ	Beschreibung
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.		
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 bearbeitete 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.
ecursiveDelete	boolean	Aktiviert das rekursive Löschen. Alle Unterseiten und deren Inhalte werden mitgelöscht

9. Anhang

Rückmeldungen

Auf Anregungen, Kritik und Lob zur Deutschen Typo3-Dokumentation freue ich mich unter der E-Mail-Adresse t3doku@mittwald.de .

Interessante Links

Diese Dokumentation befindet sich ständig in der Weiterentwicklung. Den aktuellen Stand können Sie sich unter http://www.mittwald.de/dokumentationen.html herunterladen.

Bei Fragen zu TYPO3 (Probleme mit der Entwicklung): Das Deutsche TYPO3-Forum: http://www.typo3.net

Wichtigste Quellen zum Thema TypoScript sind die TypoScript Referenze - TSRef und die TSConfig Referenz auf http://typo3.org/documentation/document-library/ .

Informationen zu Typo3-Schulungen: http://www.mittwald.de

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.mittwald.de