

Kai Laborenz
Thomas Wendt
Andrea Ertel
Prakash Dussoye
Elmar Hinz

TYPO3

Das Handbuch für Entwickler

Galileo Computing 

Auf einen Blick

1	Vorwort	15
Teil I	Grundlagen	21
2	Einführung	23
3	Grundlagen und Systemarchitektur von TYPO3	29
4	Der Aufbau von TYPO3	43
5	Die TYPO3-Community und Ressourcen zu TYPO3	53
Teil II	TYPO3 verwenden	59
6	Installation	61
7	Das Backend von TYPO3	79
8	Aufbau von TYPO3-Sites	151
9	TypoScript	241
10	Extensions anwenden	341
11	Mehrsprachige Websites mit TYPO3	391
12	Barrierefreie und standardkonforme Websites mit TYPO3	417
Teil III	TYPO3 erweitern	457
13	Eigene Extensions programmieren	459
A	CD-ROM	615
B	Zusätzliche Informationen	619
	Index	701

Inhalt

1 Vorwort 15

1.1	Für wen wurde dieses Buch geschrieben?	17
1.2	Was befindet sich in diesem Buch?	18
1.3	Was ist auf der CD-ROM und der Referenzkarte?	18
1.4	Die Website zum Buch	18
1.5	Danke!	19

Teil I Grundlagen 21

2 Einführung 23

2.1	Neu in Version 3.8	25
2.2	Pläne für die Zukunft	26
2.2.1	TYPO3 4.0 (in der Entwicklung, Veröffentlichung geplant für Ende 2005)	26
2.2.2	TYPO3 4.5 (Veröffentlichung geplant für Frühjahr 2006)	26
2.2.3	TYPO3 5.0 (Veröffentlichung geplant für Herbst 2006)	26

3 Grundlagen und Systemarchitektur von TYPO3 29

3.1	Grundlegendes	31
3.1.1	Trennung von Inhalten und Darstellung	31
3.1.2	Erweiterbarkeit	32
3.1.3	Mehrsprachigkeit	32
3.1.4	Weitere Funktionen	33
3.1.5	Entstehung von TYPO3	34
3.1.6	Community	34
3.1.7	Open Source	35
3.2	Systemaufbau	35
3.2.1	Der Systemkern von TYPO3	37
3.2.2	Erweiterungen (Extensions)	37
3.3	Konfiguration	38

3.4	Datenbank	39
3.4.1	MySQL	39
3.4.2	Verwendung anderer Datenbanken mit TYPO3	40
3.4.3	Wichtige Tabellen in der TYPO3-Datenbank	40

4 Der Aufbau von TYPO3 43

4.1	Dateinamen und Verzeichnisse	45
4.1.1	Verzeichnisse einer Standard-Installation	45
4.1.2	Wichtige Dateien	46
4.2	Voraussetzungen zur Installation von TYPO3	46
4.2.1	Hardware und Betriebssystem	46
4.2.2	Software	47
4.3	Performance	48
4.3.1	Praktische Hinweise	48
4.3.2	Performance-Steigerungen	49

5 Die TYPO3-Community und Ressourcen zu TYPO3 53

5.1	Dokumentationen	55
5.1.1	Dokumentationen für Redakteure	56
5.1.2	Quellcode-Dokumentation	56
5.2	Websites, Foren und Mailinglisten	56
5.2.1	Mailinglisten und Newsgroups	58
5.2.2	TYPO3-Websites	58

Teil II TYPO3 verwenden 59

6 Installation 61

6.1	Installation von TYPO3	63
6.2	Installation mehrerer Sites auf einem TYPO3-System	68
6.2.1	Ein Kern – viele TYPO3	68
6.2.2	Ein TYPO3 – viele Websites	69
6.2.3	Erstellen multipler Sites mit Freesite	70
6.3	Anbindung an bestehende IT-Strukturen (LDAP)	71
6.3.1	Kurzbeschreibung LDAP	72
6.3.2	Installation der LDAP-Extension	72
6.3.3	Konfiguration der LDAP-Extension	73
6.3.4	Kontaktaufnahme zum LDAP-Server	74
6.3.5	Fehlersuche	77

7 Das Backend von TYPO3 79

7.1	Die Benutzeroberfläche von TYPO3	81
7.1.1	Die Module und ihre Funktionen	83
7.1.2	Seiten anlegen und bearbeiten	87
7.1.3	Seiteninhalte einpflegen	91
7.1.4	Allgemeine Datensatzverwaltung in Root	95
7.1.5	AdminPanel	98
7.1.6	Import/Export von T3D-Dateien	104
7.1.7	Sicherung von TYPO3-Installationen	107
7.2	Anpassen des Backends über TSconfig	109
7.2.1	TSconfig	109
7.2.2	TypoScript-Wizard	110
7.2.3	Page TSconfig	111
7.2.4	User TSconfig	115
7.2.5	Beispiele zur Backend-Anpassung	117
7.3	Rich Text Editor	128
7.3.1	RTE-Transformationen	128
7.3.2	RTE-Konfiguration der Benutzeroberfläche	132
7.3.3	Die Toolbar des Rich Text Editors	132
7.3.4	pageTSconfig.txt	133
7.3.5	Absatzarten und Zeichenarten begrenzen	134
7.3.6	Klassen definieren und als Absatzart zuweisen	136
7.3.7	Ausgabekonfiguration	139
7.4	Alternative Editoren für das Backend	141
7.4.1	HTMLarea	142
7.4.2	Andere Editoren	146
7.4.3	Eine Beispielkonfiguration für den RTE (HTMLarea)	147

8 Aufbau von TYPO3-Sites 151

8.1	Vorgehensweise	153
8.1.1	Ziele und Zielgruppen festlegen	154
8.1.2	Inhalte planen	154
8.1.3	Funktionen festlegen	154
8.1.4	Dokumententypen festlegen	155
8.1.5	Sitemap/Struktur planen	155
8.1.6	Designvorlagen vorbereiten	156
8.1.7	Extensions auswählen	157
8.1.8	Template-Struktur planen	158
8.1.9	Designvorlagen einbinden	159
8.1.10	Struktur anlegen	159
8.1.11	Benutzergruppen/Benutzer	160
8.1.12	Inhalte eingeben	160

9.3.6	Basis-Templates – Eigene Standard-Templates	299
9.3.7	Templates ab der nächsten Ebene (Templates on next Level) ..	300
9.3.8	Root, Rootline und Rootline-Arrays	300
9.4	Die Template Tools	301
9.4.1	Template Tool: Info/Modify	301
9.4.2	Template Tool: Template Analyzer	303
9.4.3	Template Tool: TypoScript Object Browser	303
9.4.4	Template Tool: Constant Editor	305
9.4.5	Erstellen eigener Eingabemasken im Constant Editor	306
9.4.6	Template Tool: CSS Styler	313
9.4.7	Clear all cache – admin functions	313
9.4.8	setup.txt und constants.txt	314
9.4.9	TSconfig	314
9.5	Der Rendering-Prozess	314
9.6	TypoScript-Schnipsel/Praxisbeispiele	316
9.6.1	Allgemeine Seitenkonfiguration	316
9.6.2	TypoScript-Menüs	322
9.6.3	Navigation aus Bildern der Mediafelder	328
9.6.4	Ändern des Standard-<body>-Tag	330
9.6.5	Auswählbare Wraps um den gesamten Inhalt der Spalte NORMAL	331
9.6.6	Sitemap/Menü	333
9.6.7	Login-Formular per TypoScript	334
9.6.8	Ausblenden der erweiterten Suche der Extension »indexed search«	337
9.6.9	Anzahl der Bilder des Mediafeldes erhöhen	337
9.6.10	Mehrsprachigkeit	338

10 Extensions anwenden 341

10.1	Allgemeine Definition von Extensions	344
10.2	Einteilung von Extensions	344
10.2.1	Einteilung nach Bedeutung	344
10.2.2	Einteilung nach Funktion	345
10.3	Verwaltung von Extensions	346
10.3.1	Extension-Manager	347
10.3.2	TYPO3-Online-Repository	356
10.4	Der Extension Kickstarter Wizard	359
10.5	Anwendungsbeispiele von Extensions	360
10.5.1	Extension: News	360
10.5.2	Frontend-Anpassungen	374
10.5.3	Extension RSS	381
10.5.4	Die Extension »CSS Styled Content«	385

11 Mehrsprachige Websites mit TYPO3 391

11.1	Unicode (utf8)	393
11.1.1	Was ist Unicode?	393
11.1.2	Technische Zusammenfassung	395
11.2	TYPO3 als utf8 einrichten	396
11.2.1	Überblick	396
11.2.2	Sprachdateien, TypoScript und utf8	397
11.2.3	Grundlagen zur Zeit	397
11.2.4	Die Locales	399
11.3	TypoScript-Einstellungen für einsprachige Seiten	400
11.4	TypoScript-Einstellungen für mehrsprachige Seiten	404
11.4.1	Ein Seitenbaum für jede Sprache (Multi-Tree-Ansatz)	404
11.4.2	Ein Seitenbaum für alle Sprachen (Single-Tree-Ansatz)	405

12 Barrierefreie und standardkonforme Websites mit TYPO3 417

12.1	Warum Sie barrierefrei entwickeln sollten	419
12.2	Wie sieht eine barrierefreie Website aus?	419
12.2.1	Inhalt und Präsentation trennen	419
12.2.2	Logische Strukturen schaffen	420
12.2.3	Nicht für bestimmte Geräte arbeiten	420
12.2.4	Abwärtskompatible Seiten erstellen	420
12.2.5	Alternativen bereitstellen	421
12.2.6	Standards beachten	421
12.3	Standardkonforme Seiten mit TYPO3 erstellen	421
12.3.1	XHTML-Cleaning	422
12.3.2	Die Extension »Page Validator«	423
12.4	Barrierefreie Websites mit TYPO3	424
12.4.1	Vorhandene Hilfsmittel	424
12.5	TYPO3-Inhaltselemente barrierefrei gestalten	427
12.5.1	Überschrift und Text	427
12.5.2	Text-Bild und Bilder	428
12.5.3	Aufzählung (Punktliste)	428
12.5.4	Tabelle	429
12.5.5	Dateiliste	434
12.5.6	Multimedia	435
12.5.7	Sitemap/Menü	435
12.5.8	HTML	436
12.5.9	Mailformular	436
12.5.10	Suchformular	438
12.5.11	Anmeldeformular	439
12.5.12	Plugins	439
12.5.13	Sprungmarken à la XHTML	439

12.6	Barrierefreie Menüs mit TYPO3	440
12.6.1	Textmenüs	440
12.6.2	Grafische Menüs	443
12.7	Ein barrierefreier RTE	443
12.8	Einbindung von Tidy	444
12.9	Accessible Content	445
12.9.1	Exkurs: Rendering von Content-Elementen	446
12.9.2	Neues Rendering erstellen	448
12.9.3	Die XCLASS	449
12.9.4	Tabellen	450
12.9.5	Formulare	452
12.9.6	Probleme mit Plugins	454
12.9.7	Weitere Entwicklung von »Accessible Content«	455

Teil III TYPO3 erweitern 457

13 Eigene Extensions programmieren 459

13.1	Grundlagen	461
13.1.1	Namenskonventionen	461
13.1.2	Dateistruktur einer Extension	462
13.1.3	Der Extension-Manager	463
13.1.4	Die Datei ext_emconf.php	464
13.1.5	Die Datei ext_localconf.php	467
13.1.6	Die Datei ext_tables.php	468
13.2	Drei Extensions als Beispiel	468
13.2.1	Die Extension »Text w/ summary«	468
13.2.2	Die Extension »Function Reference«	470
13.2.3	Die Extension »Akronymmanager«	472
13.3	Globale Variablen und Datenbankfunktionen	473
13.3.1	Variablen	473
13.3.2	Datenbankzugriffe mit dem DBAL	475
13.3.3	Datenbanktabellen	478
13.4	Backend-Programmierung	480
13.4.1	Eingabemasken	481
13.4.2	Das Table Configuration Array \$TCA	482
13.4.3	\$TCA-Beispiel: Extension »Function Reference«	484
13.4.4	\$TCA-Beispiel: Wizard austauschen	493
13.4.5	Context Sensitive Help (CSH)	494
13.4.6	TCE und TCEforms	498
13.4.7	Anlegen und Bearbeiten von Inhaltstypen	501
13.4.8	Der Inhaltstyp »Plugin einfügen«	504
13.5	Backend-Module	509
13.5.1	Anlegen eines Moduls	509
13.5.2	Ausgabeformatierungen in Modulen	512
13.5.3	Einfügen eines Submoduls in das Menü	516

13.5.4	Beispiel für Module – der Akronymmanager	518
13.5.5	Der Navigationsbereich	523
13.5.6	Rechte von Backend-Usern abfragen	524
13.6	Frontend-Programmierung	526
13.6.1	Das Frontend-Rendering	526
13.6.2	Objekte im \$TSFE	528
13.6.3	Standard-Rendering-Funktionen	528
13.6.4	Ausgabe eigener Inhalte mittels USER, USER_INT, PHP_SCRIPT, PHP_SCRIPT_INT oder PHP_SCRIPT_EXT	530
13.6.5	Neue Rendering-Funktionen anlegen	531
13.6.6	stdWrap in Rendering-Funktionen	534
13.6.7	Frontend-User und Sessions	534
13.6.8	Eingabeformulare	535
13.6.9	Beispiel für Eingabeformulare – Pinboard für die »Function Reference«	537
13.6.10	HTML-Vorlagen	543
13.6.11	CSS und JavaScript	546
13.7	Plugins – die Klasse tslib_pibase	546
13.7.1	tslib_pibase – Link-Methoden	547
13.7.2	tslib_pibase – Methoden für das Auflisten von Datensätzen	549
13.7.3	tslib_pibase – Methoden für Stylesheet und CSS	552
13.7.4	tslib_pibase – Methoden für das Frontend-Editing	553
13.7.5	tslib_pibase – Methoden für Mehrsprachigkeit und Lokalisation	555
13.7.6	tslib_pibase – Methoden für Datenbankabfragen	556
13.7.7	tslib_pibase – Methoden für Flexforms	557
13.8	Flexforms	558
13.8.1	Hello World mit Flexforms	559
13.8.2	Flexform einbinden	563
13.8.3	Flexforms gestalten	566
13.8.4	Flexform – Grundstruktur	566
13.8.5	Flexform-Input-Felder	567
13.8.6	Flexform dynamisch	582
13.8.7	Auf Flexform-Daten zugreifen	582
13.9	Mehrsprachigkeit	584
13.9.1	Eigene Locallang-Dateien für einzelne Sprachen	588
13.9.2	Extensions für die Arbeit mit Locallang-Dateien	588
13.10	Allgemeines	588
13.10.1	TypoScript in Extensions	589
13.10.2	Static-Templates hinzufügen	589
13.10.3	TypoScript einfügen	590
13.10.4	Klassen erweitern – XCLASS	590
13.10.5	Hooks	592
13.10.6	Services	595
13.10.7	Sicher programmieren	597
13.10.8	Der Extension Kickstarter	599
13.10.9	Der Development Evaluator	602
13.10.10	Dokumentation	604
13.11	Caching-FAQ	607

A	CD-ROM	615
A.1	Dateien zum Buch	615
A.2	Dokumentationen	615
A.3	Videos	616
A.4	TYPO3 3.8 zum Installieren	616
A.5	Extensions	616
A.6	TYPO3_API_Doxygen	617
B	Zusätzliche Informationen	619
B.1	Das Install-Tool und seine Optionen	619
B.1.1	Basic Configuration	619
B.1.2	Database Analyzer	622
B.1.3	Image Processing	623
B.1.4	All Configuration	625
B.2	Ein reines CSS-Menü (für Kapitel 12)	639
B.3	CSS Styled Content im Detail	643
B.3.1	CONSTANTS: EXT:css_styled_content/static/	643
B.3.2	SETUP: EXT:css_styled_content/static/	646
	Index	701

2 Einführung

TYPO3 hat sich in den Jahren seiner Existenz stetig weiterentwickelt und verbessert. Einige der wichtigsten Änderungen sind in den letzten drei Versionssprüngen (ab Version 3.6) entstanden, und für die nahe Zukunft sind neue Meilensteine angekündigt.

2.1 Neu in Version 3.8

Der Umstieg von TYPO3 3.7 auf 3.8 hat eine Reihe von Veränderungen und Verbesserungen gebracht:

- ▶ Statt ImageMagick kann nun GraphicsMagick (<http://www.graphicsmagick.org/> (**Linkcode 002**)) zur Bildmanipulation verwendet werden – es soll schneller sein, weniger Ressourcen verbrauchen und bessere Bildqualität liefern (optional im Install-Tool wählbar).
- ▶ Spracheinstellungen als Extensions
- ▶ Verwendung von Cache-Control-Headern zur Performance-Steigerung (siehe Abschnitt 4.3.1, *Performance-Steigerungen*)
- ▶ Neuer Ordner Systemextensions (**typo3/sysext/**) für grundlegende Extensions
- ▶ Verbesserungen in der Backend-Bedienung (u.a. funktioniert das Kontextmenü auch in Opera, Backend-Module können zur besseren Übersicht eingeklappt werden)
- ▶ Verbesserungen im Frontend (Frontend-Rendering kann nun auf Teile des Seitenbaums beschränkt werden, Frontend-Loginfelder sind »case-sensitiv«, und Cookies für Frontend-Benutzer können mit einer Lebensdauer versehen werden; die Spam-Verschlüsselung für E-Mails funktioniert jetzt ohne Javascript.)
- ▶ Verbesserungen der Extension »Indexed Search« (Indizierung von Excel, Powerpoint und OpenOffice, Statistiken über die Suchwörter; leider noch keine Templates oder tabellenfreier Code für die Ausgabe)
- ▶ Umfangreiche Verbesserungen der »Import/Export«-Extension
- ▶ Auch an der Datenbankabstraktion (Verwendung anderer Datenbanken als MySQL) wurde gearbeitet (siehe dazu Abschnitt 13.3.2, *Datenbankzugriffe mit dem DBAL*).

Allerdings sind einige sehnlichst erwartete Verbesserungen noch nicht offiziell dabei:

- ▶ TemplaVoila
- ▶ Digital Asset Management (DAM)
- ▶ Workflow-Management

Diese und andere Funktionsblöcke sollen jedoch mit der kommenden Version TYPO3 4.0 integriert werden – lesen Sie dazu das nächste Kapitel.

2.2 Pläne für die Zukunft

Die TYPO3-Community hat einige Pläne für die Zukunft – laut der aktuellen Roadmap können wir folgende zukünftige Entwicklungen erwarten:

2.2.1 TYPO3 4.0 (in der Entwicklung, Veröffentlichung geplant für Ende 2005)

- ▶ PHP 4- und PHP 5-Kompatibilität
- ▶ Workflow-Management
- ▶ Versionsverwaltung
- ▶ Workspaces 1.0
- ▶ Digital Asset Management 1.0
- ▶ Weiterentwicklung des Extension Managers
- ▶ Neuer Standard-RTE
- ▶ TemplaVoila 1.0
- ▶ Datenbankabstraktionsmodell (DBAL 1.0)
- ▶ Indexed Search 2.0 (mit Templates)
- ▶ Methoden zur Prüfung und Verbesserung der Datenintegrität
- ▶ Verbesserung der Bedienungsfreundlichkeit

2.2.2 TYPO3 4.5 (Veröffentlichung geplant für Frühjahr 2006)

- ▶ Redesign des Backends
- ▶ Weitere Verbesserung der Bedienungsfreundlichkeit

2.2.3 TYPO3 5.0 (Veröffentlichung geplant für Herbst 2006)

- ▶ PHP5-Optimierungen
- ▶ Verbesserungen der Systemarchitektur

- ▶ Portalfunktionen
- ▶ Änderungen im TYPO3-Kern zur Verbesserung der Bedienungsfreundlichkeit
- ▶ Integration von SOAP
- ▶ WebDAV

12 Barrierefreie und standardkonforme Websites mit TYPO3

In den letzten Jahren hat die Bedeutung standardkonformer und barrierefreier Websites enorm zugenommen. Der Grund dafür ist in Deutschland unter anderem die »Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz« (BITV), die barrierefreie Webauftritte für Bundesbehörden und deren Körperschaften bis spätestens Ende 2005 vorschreibt.

12.1 Warum Sie barrierefrei entwickeln sollten

Aber auch das gestiegene Bewusstsein, dass das WWW in erster Linie ein Informationsmedium ist, und die zunehmende Anzahl von mobilen Internet-Clients lässt immer mehr Kunden nach Barrierefreiheit verlangen.

Wir verwenden hier der Verständlichkeit halber den eingeführten Begriff »Barrierefreiheit«, obwohl genau genommen eher von »Barrierearmut« gesprochen werden muss. Aufgrund der vielen möglichen unterschiedlichen Einschränkungen ist es kaum möglich, eine Website so zu erstellen, dass sie wirklich keine Barrieren für niemanden mehr enthält.

12

Die Vorteile einer barrierefreien, standardkompatiblen Website sind:

- ▶ zugänglich für alle Menschen
- ▶ gut indizierbar für Suchmaschinen
- ▶ leicht zu warten
- ▶ geringe Dateigrößen – schnell zu laden – geringe Traffic-Kosten
- ▶ auch auf mobilen Geräten darstellbar
- ▶ zukunftssicher

12.2 Wie sieht eine barrierefreie Website aus?

Was ist nun notwendig, um eine barrierefreie Website zu erreichen? Sehen wir uns die Anforderungen zunächst allgemeiner an, um dann auf die konkreten technischen Kriterien zu kommen.

12.2.1 Inhalt und Präsentation trennen

Der HTML-Code enthält die Dokumentenstruktur – das Stylesheet die Layout- und Designinformationen. Nur so können Geräte, die mit der optischen Darstellung nichts anfangen können, die Struktur zur Präsentation verwenden.

12.2.2 Logische Strukturen schaffen

Verwenden Sie die HTML-Tags in ihrer Funktion als semantische Gliederungsbefehle: `<h1>` markiert die wichtigste Überschriftenebene eines Dokuments, und `<table>` leitet eine tabellarische Darstellung von Daten ein – nicht ein mehrspaltiges Dokument.

Ein nicht unerheblicher Aufwand entsteht durch die Forderung, Abkürzungen und Sprachwechsel im Dokument zu markieren.

Um so schlimmer ist es, dass der Internet Explorer gerade mit dem Element `<abbr title="..."> ... </abbr>` (für Abkürzungen) nichts anfangen kann. Weder zeigt er den Titel (also die ausgeschriebene Form) an noch wendet er per CSS zugewiesene Stile an. Eine Lösung hat der tschechische Entwickler Marek Prokop auf seiner Website <http://www.sovavsiti.cz/css/abbr.html> (**Linkcode 045**) entwickelt (keine Angst, der Artikel ist in englischer Sprache verfasst). In der nächsten Version von XHTML wird es im Übrigen auch nur noch das Element `<abbr>` geben – hoffen wir, dass auch der Internet Explorer es bis dahin gelernt hat.

12.2.3 Nicht für bestimmte Geräte arbeiten

Denken Sie bei der Planung der Seiten nicht nur an grafische Browser – und vor allem nicht an einen bestimmten Browser, ein bestimmtes Betriebssystem oder eine bestimmte Bildschirmauflösung. Sie können letztendlich nicht wissen, wer wo mit welchem Gerät Ihre Website besucht. Verzichten Sie auf proprietäre Befehle, auch wenn sie noch so verführerisch erscheinen.

Verwenden Sie möglichst relative Größenangaben statt absoluter, damit sehgeschwache Nutzer die Schrift vergrößern können.

12.2.4 Abwärtskompatible Seiten erstellen

Wenn Sie dennoch besondere Features nutzen (müssen), die nur auf neuen oder bestimmten Geräten funktionieren, sorgen Sie dafür, dass die Seiten auch ohne diese Features funktionieren.

Wenn Sie eine Navigationsleiste mit einem Java-Applet oder durch eine Flash-Animation realisieren, zeigt ein Browser ohne Java oder Flash-Plugin *gar nichts* an. Würde die Navigationsleiste aus Grafiken bestehen, die mit einem Mouse-Over-Effekt versehen sind, so könnten selbst Personen, die ihre Grafiken abgeschaltet haben, noch die alternativen Texte sehen (natürlich nur, sofern Sie diese angegeben haben).

12.2.5 Alternativen bereitstellen

Wenn Sie Inhalte oder Funktionen verwenden, die in Hinsicht auf die Zugänglichkeit problematisch sind, stellen Sie Alternativen bereit.

Grafiken können abgeschaltet werden bzw. sind Blinden nicht zugänglich: Verwenden Sie immer das `alt`-Attribut des ``-Tags.

Bieten Sie zu einer auf JavaScript oder Flash basierenden Navigation eine alternative Text-Navigation an. Stellen Sie für Videos, Animationen oder gesprochene Texte eine Beschreibung bzw. ein Transkript zur Verfügung. Ergänzen Sie mausspezifische Ereignisse um entsprechende Tastatur-Ereignisse: Zu jedem `:hover` gehört ein `:focus`.

12.2.6 Standards beachten

Beachten Sie die Standards des W3C, und erstellen Sie Webseiten in HTML 4 oder XHTML 1. Testen Sie Ihre Dokumente mit den vom W3C bereitgestellten Validatoren.

Detailliertere Informationen, was zum Erreichen einer barrierefreien Website erforderlich ist, finden Sie in der BITV, den Kriterien des BIK-Kurztests oder im Kriterienkatalog des BIENE-Wettbewerbs der Aktion Mensch (<http://www.biene-award.de/> (Linkcode 046)).

Sie finden diese Unterlagen auf der dem Buch beiliegenden CD-ROM.

12.3 Standardkonforme Seiten mit TYPO3 erstellen

Das Erstellen standardkonformer Seiten mit TYPO3 ist in den Varianten *HTML 4.01* und *XHTML 1.0 Transitional* vergleichsweise einfach möglich.

Um die TYPO3-StandardEinstellung von *HTML 4.01* und *XHTML 1.0 Transitional* umzustellen, ist lediglich folgende Zeile im Root-Template erforderlich:

```
page.config.doctype = xhtml_trans
```

Generell sind folgende Werte möglich:

Wert für <code>page.config.doctype</code>	Ausgegebener Doctype
<code>xhtml_trans</code>	XHTML 1.0 Transitional
<code>xhtml_frames</code>	XHTML 1.0 Frameset
<code>xhtml_strict</code>	XHTML 1.0 Strict

Tabelle 12.1 Doctypes und TypoScript-Konfiguration

Wert für <code>page.config.doctype</code>	Ausgegebener Doctype
<code>xhtml_11</code>	XHTML 1.1
<code>xhtml_20</code>	XHTML 2
<code>none</code>	Gar kein Doctype

Tabelle 12.1 Doctypes und TypoScript-Konfiguration

Achtung Dies steuert nur die Ausgabe des Doctypes – es führt nicht zu einer entsprechenden Ausgabe des HTML-Codes!

Zudem gibt es noch die Möglichkeit, den benötigten XML-Prolog zu steuern:

```
page.config.xmlprologue = none
```

schaltet die Ausgabe des Prologs ab – das ist notwendig, damit der Internet Explorer in den Standard-Modus geschaltet wird (zum Thema Rendering-Modi lesen Sie <http://www.fabrice-pascal.de/artikel/dtd> (Linkcode 047)).

12.3.1 XHTML-Cleaning

Die Konfigurationsoption:

```
config.xhtml_cleaning = all
```

schaltet eine Art XHTML-Säuberung ein, die allerdings noch nicht wirklich vollständig ist. Das Aktivieren der Option sorgt dafür, dass

- ▶ Element- und Attributnamen kleingeschrieben werden,
- ▶ Einzelelemente (`` oder `
`) geschlossen werden,
- ▶ Anführungszeichen um alle Attribute gesetzt werden,
- ▶ ALT-Attribute in IMG-Elemente gesetzt werden, wenn diese fehlen.

Weitergehende Erfordernisse von XHTML, wie die korrekte Verschachtelung oder die CDATA-Umschließung für Skripte, werden allerdings noch nicht berücksichtigt.

Als Option kann gesteuert werden, wann der Code durch den »Cleaner« geschickt wird:

- ▶ `all` – immer
- ▶ `cached` – wenn die Seite in den Cache geschrieben wird
- ▶ `output` – bevor die Seite ausgegeben wird

Des Weiteren existiert eine Extension »Advanced HTML Cleaner« (http://typo3.org/extensions/repository/search/qcom_htmlcleaner/ (Linkcode 048)), die weitere Verbesserungen des Codes ermöglicht. So werden je nach Doctype unerlaubte Elemente und Attribute entfernt oder verändert. Durch Anpassen des Quellcodes der Datei `class.ux_t3lib_parsehtml.php` (`/typo3conf/ext/qcom_htmlcleaner/`) können hier auch eigene Umschreibungen definiert werden. Die Verwendung erfordert das Aktivieren von `config.xhtml_cleaning`.

Ein Problem bei der Realisierung valider Websites mit TYPO3 sind Extensions, die oft nur mit erheblichen Änderungen im PHP-Code zu standardkonformer Ausgabe gebracht werden können. Bei restriktiveren Doctypes wie *XHTML 1.0 strict* und *XHTML 1.1* verschärfen sich diese Probleme noch.

12.3.2 Die Extension »Page Validator«

Die Extension »Page Validator« (`sf_validator`) hilft, die Validität von Seiten zu überprüfen. Mit einem Klick zeigt sie die Ergebnisse einer Prüfung mit dem Validator des W3C.

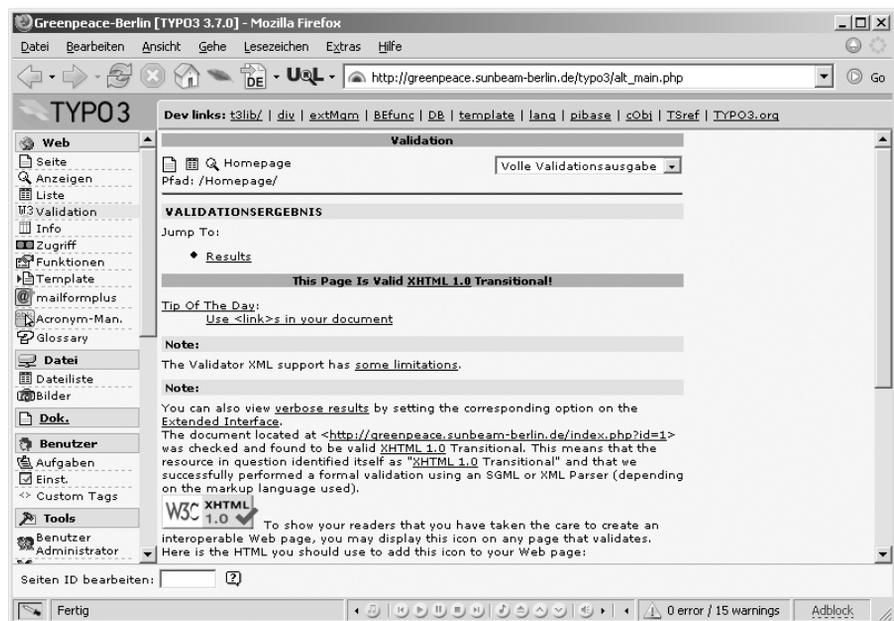


Abbildung 12.1 Gratulation vom W3C – diese Seite ist valide!

Installation und Bedienung sind einfach: Laden Sie die Extension aus dem Online-Repository herunter (Extension Key: `sf_validator`), und installieren Sie sie. Danach steht im Modul **Web** (nach dem Neuladen der Seite) das Submodul

Validation zur Verfügung. Zum Validieren muss die betreffende Seite nur noch angewählt werden. Über das Funktionsmenü können noch einige wenige Einstellungen vorgenommen werden – z.B. wie ausführlich die Kommentare des Validators angezeigt werden.

12.4 Barrierefreie Websites mit TYPO3

Die Probleme beim Erstellen barrierefreier Seiten liegen bei TYPO3 vor allem in folgenden Bereichen:

1. Es fehlt die Möglichkeit, typische Anforderungen an Barrierefreiheit über das Backend zu realisieren. So gibt es im »Text-Bild«-Objekt standardmäßig keine Möglichkeit, einen alternativen Bildtext (`alt`-Attribut) oder eine Langbeschreibung (`longdesc`) einzugeben. Das Objekt »Tabelle« lässt so wichtige Dinge wie die `<caption>` vermissen oder die Möglichkeit, Tabellenkopfzeilen als solche zu markieren.
2. Die Inhaltselemente werden weitgehend über komplizierte Tabellenkonstruktionen positioniert. Neuere Extensions wie »CSS Styled Content« beginnen zwar mit einem mehr CSS-orientierten Layout, dies ist aber bei weitem noch nicht vollständig.
3. Einige häufig verwendete Extensions (z.B. »Indexed Search«) sind ebenfalls noch auf Basis von Tabellenlayouts realisiert.

Zusätzlich animieren die ausgezeichneten Grafikfunktionen von TYPO3 viele Designer dazu, rein grafische Menüs zu entwerfen – auch kein Plus für barrierefreie Sites, da grafische Texte nicht vergrößert werden können.

12.4.1 Vorhandene Hilfsmittel

Neben den Einschränkungen gibt es für TYPO3 auch einige Werkzeuge, die beim Erstellen barrierefreier Seiten helfen.

Der erste Ansatz, Gestaltung über CSS zu realisieren, war das statische Template »cSet stylesheet«. Dieses ist inzwischen aber durch »CSS Styled Content« überholt und sollte nicht mehr verwendet werden.

Inzwischen existiert eine Reihe von Extensions, die sich der Aufgabe widmen, TYPO3 mehr Barrierefreiheit beizubringen.

CSS Styled Content (`css_styled_content`)

Die wichtigste verfügbare Erweiterung für barrierearmen Code ist »CSS Styled Content« (`css_styled_content`), mit der ein tabellenfreier Code erzeugt wird – zumindest in einigen Bereichen.

»CSS Styled Content« setzt im Wesentlichen TypoScript-Vorgaben für die Inhaltselemente ein, die auf Font-Elemente zur Gestaltung verzichten (im Gegensatz zu »Content (default)«, und schreibt die Funktionen für die Inhaltselemente

- ▶ Bullelist (wird als ungeordnete HTML-Liste ausgegeben)
- ▶ Table
- ▶ Dateiliste (wird allerdings immer noch als Tabelle ausgegeben)

in `/tslib/tslib_piibase` neu.

Leider versetzt uns »CSS Styled Content« allein noch nicht in die Lage, barrierefreie Websites zu erstellen. Es berücksichtigt nicht alle Inhaltselemente, insbesondere nicht das wichtige »Text mit Bild«-Objekt.

Außerdem wird aus Gründen der Verallgemeinerbarkeit ein wahrer »Klassen-Overkill« erzeugt, der viele Elemente mit für die meisten Anwendungsbereiche unnötig vielen Klassen oder nicht sinnvollen Attributen ausstattet.

Trotzdem ist es sinnvoll, sich mit »CSS Styled Content« ausführlicher zu befassen. Die dort eingesetzten Methoden lassen sich für eigene Projekte abwandeln oder zeigen, wie eigene Accessibility-Ergänzungen entwickelt werden können.

Daher haben wir im Anhang B.3 die Erweiterung im Detail unter die Lupe genommen und den Quellcode Zeile für Zeile kommentiert.

CSS Styled Imagetext (`css_styled_imgtext`)

Eine große Lücke in »CSS Styled Content« soll die Erweiterung »CSS Styled Imagetext« (CSI) stopfen: die Realisierung eines »Text mit Bild«-Objekts ohne Layouttabellen.

Die Erweiterung befindet sich allerdings noch im Stadium »Experimental«.

Dazu gehört dann auch gleich die Extension »CSS styled IMGTEXT with alt and title attributes« (`sl_css_imgtext`), die `alt`-Texte und `title` nachrüstet.

Accessible XHTML Template (`gov_accessibility`)

Dieses Template ist eine in weiten Teilen anpassbare, fertige barrierefreie Site – eine Art lebende »Best-Practice-Studie«.

Nach Installation der Erweiterung steht ein komplettes Website-Layout mit Features wie Schriftgrößen- und Layoutauswahl zur Verfügung. Durch Verwendung des Objektbrowsers können Farben, Bilder und weitere Eigenschaf-

ten angepasst werden. Die Extension nutzt weitere Extensions wie »CSS Styled Content« und »CSS Styled Imagetext«.

http://typo3.org/documentation/document-library/gov_accessibility/
(Linkcode 049)

Accessible Tables (accessible tables)

Diese neue Erweiterung befasst sich mit dem Tabellenelement und ergänzt es um Barrierefreiheitsfunktionen – mehr dazu finden Sie im folgenden Abschnitt bei den Inhaltselementen.

Gov Textmenu und gov_accesskey

Diese beiden Extensions rüsten das TMENU um die Attribute `accesskey` und `tabindex` nach.

Allerdings wird die Verwendung dieser Attribute von Accessibility-Spezialisten inzwischen nicht mehr empfohlen (siehe <http://www.einfach-fuer-alle.de/artikel/tabindex/> (Linkcode 050) und <http://www.einfach-fuer-alle.de/artikel/formulare/tag4/> (Linkcode 051)). Daher kann auf den Einsatz dieser Extensions verzichtet werden.

Akronymmanager

Der »Akronymmanager« (*sb_akronymmanager*) ist ein Hilfsmittel, um die für Barrierefreiheit erforderlichen Auszeichnungen von Abkürzungen (`<abbr>`), Akronymen (das sind Abkürzungen, die als zusammenhängendes Wort gesprochen werden, wie z.B. UNO – als `<acronym>` zu kennzeichnen) und Sprachwechseln zu erleichtern.

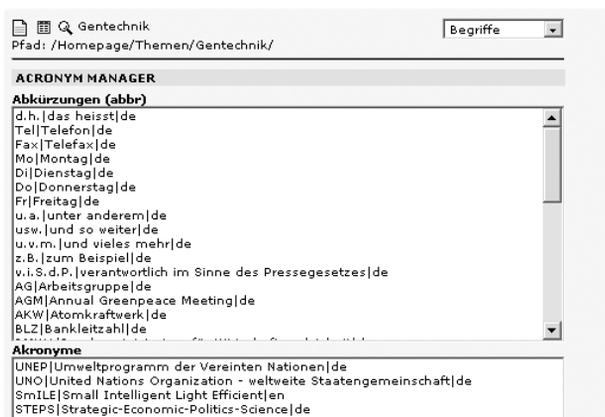


Abbildung 12.2 Der Akronymmanager im Einsatz

Über ein neues Backend-Modul können in drei Feldern Begriffe eingegeben werden, die dann bei der Ausgabe vollautomatisch mit den passenden HTML-Tags umschlossen werden.

Sie finden den Acronymmanager auf der CD-ROM, im Online-Repository oder unter <http://typo3.sunbeam-berlin.de/acronymmanager/> (Linkcode 052).

Die Extension befindet sich im Stadium »alpha« und ist noch nicht voll funktionsfähig. So werden z.B. keine Elemente zusammengefasst (wenn ein fremdsprachiges Wort allein verlinkt ist, sollten die lang-Attribute direkt in das Linkelement geschrieben werden). Die Extension wird jedoch ständig weiterentwickelt.

12.5 TYPO3-Inhaltselemente barrierefrei gestalten

12.5.1 Überschrift und Text

Verhältnismäßig unproblematisch sind die Elemente »Überschrift« und »Text«. Hier sind vor allem Anpassungen am TypoScript und eine sinnvolle Konfiguration des RTEs erforderlich.

Zusätzlich müssen noch die **spacer.gif**-Dateien entfernt werden, die beim Eintragen von Abständen nach oben oder unten (1) erzeugt werden. Diese könnten z.B. durch ein `<hr>`-Element mit per Style zugewiesener Höhe ersetzt werden.

Abbildung 12.3 Überschrift- und Textelemente sind verhältnismäßig problemlos.

Funktionen wie »Rahmen« (2), »Nach oben«-Link (3) und »Datum« (4) werden bereits durch »CSS Styled Content« angepasst (wenn auch etwas überschwänglich mit Klassen versehen).

Die Konfiguration des (bzw. eines) Editors (RTE) ist etwas kniffliger, und letztlich hängt die Konfiguration von den konkreten Einsatzbedingungen ab.

1. Alle Bedienbuttons, die problematische Elemente realisieren, sollten ausgeblendet werden, z.B. die Buttons zur Schriftformatierung per FONT-Tag.
2. Zusätzlich können passende Klassen und benutzerdefinierte Tags definiert werden (z.B. `<cite>` für Zitate).

Der Konfiguration des RTE haben wir ein eigenes Kapitel gewidmet – lesen Sie dazu in den Abschnitten 7.3, *Rich Text Editor*, und 7.4.3, *Eine Beispiel Konfiguration für den RTE*, weiter.

12.5.2 Text-Bild und Bilder

Ein größeres Problem ist der Umgang mit Bildern und Text-Bild-Kombinationen. Hier setzt TYPO3 standardmäßig auf komplexe Tabellenkonstruktionen, die auch in »CSS Styled Content« nicht barrierefrei sind. Zudem fehlen im Backend Eingabemöglichkeiten für ALT-Texte oder LONGDESC-Beschreibungen.

Ein Versuch, die Formatierung auf modernes CSS-Layout umzustellen, ist die Erweiterung »CSS styled IMGTEXT« (*css_styled_imgtext*) von Ingmar Schlecht. Hier werden Bilder konsequent ohne Tabellen positioniert. Allerdings ist die Erweiterung noch im Stadium »Experimental« und kann nur eingeschränkt eingesetzt werden.

Für die notwendigen ALT-Texte gibt es ebenfalls Extensions – »Alttext for Images« (*dmc_image_alttext*) und »CSS styled IMGTEXT with alt and title attributes« (*sl_css_imgtext*) bei Verwendung von »CSS styled IMGTEXT«.

Eine andere Methode ist die Verwendung einer eigenen Erweiterung, um die Ausgabe von Bild-Text-Elementen anzupassen.

Auf der CD-ROM finden Sie die von uns erstellte Erweiterung »Accessible Content«, die auf »CSS Styled Content« basiert, aber die verschiedene Inhaltselemente anders aufbaut. Bild-Text-Elemente werden hier je nach Einsatz als einfaches IMG-Objekt, DIVs oder Definitionslisten realisiert (Definitionslisten sind eine interessante Methode, um Bild und Bildunterschrift semantisch korrekt abzubilden).

12.5.3 Aufzählung (Punktliste)

Für Aufzählungen sieht HTML die Elemente `<u1>` oder `` vor. In »CSS Styled Content« wird die Punktliste semantisch korrekt und barrierefrei ausgegeben.

12.5.4 Tabelle

Für barrierefreie Tabellen sind eine Reihe von Hilfen vorgesehen: So müssen Tabellenköpfe als `<th>` gekennzeichnet sein, und es muss eine Zusammenfassung verfügbar sein. Tabelle 12.2 zeigt erforderliche Eigenschaften von barrierefreien Tabellen.

HTML-Element/Attribut	Funktion
<code><caption></code>	Überschrift der Tabelle
Verwendung von <code><th></code>	Markiert eine Zelle als Kopfzelle.
<code>headers="ID"</code>	Attribut für <code><td></code> Ordnet eine Datenzelle der Kopfzelle ID zu.
<code>scope="col", scope="row"</code>	Attribut für <code><th></code> Ordnet einer Kopfzelle eine Spalte oder Zeile von Datenzellen zu.
<code>scope="colgroup", scope="rowgroup"</code>	Attribute für <code><th></code> Ordnet eine Überschrift mehreren untergeordneten Überschriften zu.
<code>summary</code>	Attribut für <code><table></code> Zusammenfassung der Tabelleninformation
<code><thead>, <tfoot>, <tbody></code>	Unterteilen die Tabelle in Kopf, Fußzeile und Rumpf; <code><thead></code> oder <code><tfoot></code> können auch wegfallen; <code><tbody></code> ist obligatorisch, wenn <code><thead></code> oder <code><tfoot></code> verwendet werden. Die Bereiche müssen in der nebenstehenden Reihenfolge notiert werden.

Tabelle 12.2 Elemente barrierefreier Tabellen

Eine barrierefreie Tabelle kann z.B. so aussehen:

	Spalte 1	Spalte 2	
		Subspalte 2	Subspalte 3
Reihe 1			
Reihe 2			

Abbildung 12.4 Barrierefreie Tabelle in normalem Browser

[table summary="Diese Tabelle enthält Beispieldaten"]			
[caption] Barrierefreie Tabelle [/caption]			
[td NO headers! id="leer"]	[th id="spalte1" scope="col"] Spalte 1	[th id="spalte2" scope="colgroup"] Spalte 2	
		[th id="spalte2-1" scope="col"] Subspalte 2	[th id="spalte2-2" scope="col"] Subspalte 3
[th id="reihe1" scope="row"] Reihe 1	[td headers="reihe1 spalte1"]	[td headers="reihe1 spalte2-1"]	[td headers="reihe1 spalte2-2"]
[th id="reihe2" scope="row"] Reihe 2	[td headers="reihe2 spalte1"]	[td headers="reihe2 spalte2-1"]	[td headers="reihe2 spalte2-2"]

Abbildung 12.5 Besondere Accessibility-Funktionen sichtbar gemacht

Die gezeigte Tabelle wird mit folgendem HTML-Code erzeugt:

```

1: <table border="1" summary="Diese Tabelle enthält
   Beispieldaten">
2:   <caption>Barrierefreie Tabelle</caption>
3:   <thead>
4:     <tr>
5:       <td rowspan="2" id="leer">&nbsp;</td>
6:       <th rowspan="2" id="spalte1" scope="col">Spalte 1
       </th>
7:       <th colspan="2" id="spalte2" scope="colgroup">Spalte 2
       </th>
8:     </tr>
9:     <tr>
10:      <th scope="col" id="spalte2-1">Subspalte 2</th>
11:      <th scope="col" id="spalte2-2">Subspalte 3</th>
12:    </tr>
13:  </thead>
14:  <tbody>
15:    <tr>
16:      <th id="reihe1" scope="row">Reihe 1</th>
17:      <td headers="reihe1 spalte1">&nbsp;</td>
18:      <td headers="reihe1 spalte2-1">&nbsp;</td>
19:      <td headers="reihe1 spalte2-2">&nbsp;</td>
20:    </tr>
21:    <tr>
22:      <th id="reihe2" scope="row">Reihe 2</th>
23:      <td headers="reihe2 spalte1">&nbsp;</td>

```

```

24:         <td headers="reihe2 spalte2-1">&nbsp;</td>
25:         <td headers="reihe2 spalte2-2">&nbsp;</td>
26:     </tr>
27: </tbody>
28: </table>

```

Listing 12.1 Quelltext einer barrierefreien Tabelle

Das Standard-Element für Tabellen bietet keine Möglichkeiten, diese Angaben im Backend vorzusehen.

Um Tabellen zu realisieren, stehen in TYPO3 neben dem Standard-Tabellenmodul verschiedene Extensions zur Verfügung. Einige davon eignen sich auch, um barrierefreie Tabellen herzustellen.

Extended Table Backend (mit Extended Table Service)

Ein komfortables Interface für die Konstruktion auch komplexer Tabellen bietet das »Extended Table Backend« (*th_exttable*). Hier kann über einen komplett neu geschriebenen Wizard eine Tabelle auch mit spalten- oder zeilenübergreifenden Zellen gestaltet werden. Die Bedienung ist zwar etwas ungewohnt und stellenweise ein wenig umständlich, aber dafür gibt es reichhaltige Konfigurationsmöglichkeiten für einzelne Zellen oder die gesamte Tabelle.

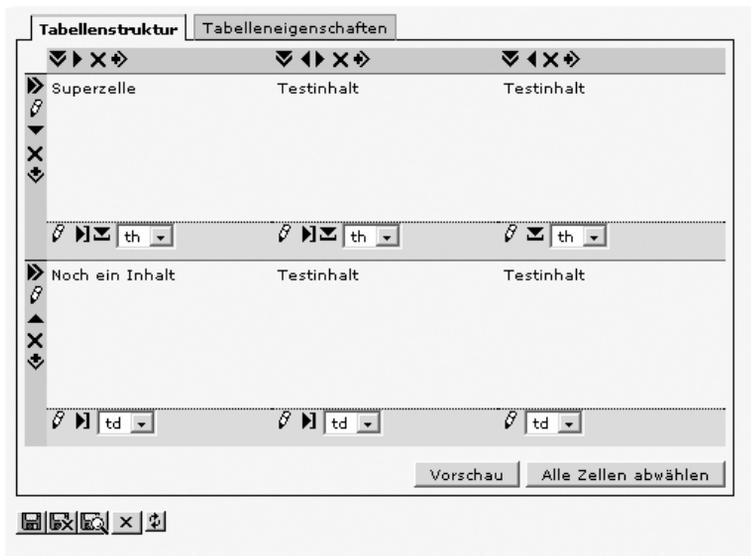
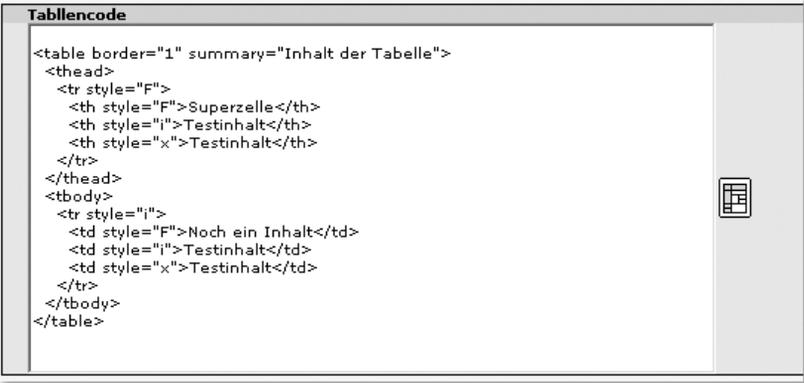


Abbildung 12.6 Ein komfortabler Editor macht auch komplexe Tabellen möglich.

Leider fehlen die meisten Funktionen, die eine barrierefreie Tabelle benötigt: `scope` und `headers` wird bislang nicht unterstützt, und auch `caption` ist noch nicht möglich (steht aber laut Autor auf der To-Do-Liste).

Da der Editor den kompletten Tabellencode in das Inhaltselement schreibt, lassen sich die fehlenden Attribute dort immerhin »per Hand« nachrüsten.



```
<table border="1" summary="Inhalt der Tabelle">
<thead>
<tr style="F">
<th style="F">Superzelle</th>
<th style="I">Testinhalt</th>
<th style="X">Testinhalt</th>
</tr>
</thead>
<tbody>
<tr style="I">
<td style="F">Noch ein Inhalt</td>
<td style="I">Testinhalt</td>
<td style="X">Testinhalt</td>
</tr>
</tbody>
</table>
```

Abbildung 12.7 Der fertige Quelltext kann per Hand nacheditiert werden.

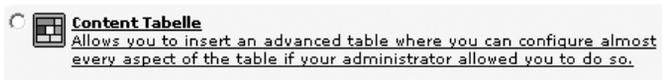


Abbildung 12.8 Ein neues Inhaltselement ermöglicht das Einfügen komplexer Tabellen.

Die Erweiterung ist als »beta« gekennzeichnet.

KB Content Table

Die Erweiterung »KB Content Table« (*kb_conttable*) bietet einen auf Flexforms basierenden Tabelleneditor, der alle nur denkbaren Optionen zur Formatierung der Tabelle und einzelner Zellen bietet (siehe Abbildung 12.9).

Sie können dabei die Editierrechte detailliert für Redakteure festlegen und einschränken. Mit dieser Erweiterung sind auch barrierefreie Tabellen möglich – jedoch ist vorher einiges an Konfigurationsarbeit nötig –, allerdings liefert Bernhard Kraft eine ausführliche Dokumentation mit, die Sie unter http://typo3.org/documentation/document-library/kb_conttable/ (Linkcode 053) finden.

Die Erweiterung ist als »alpha« spezifiziert und erfordert TYPO3 3.8.

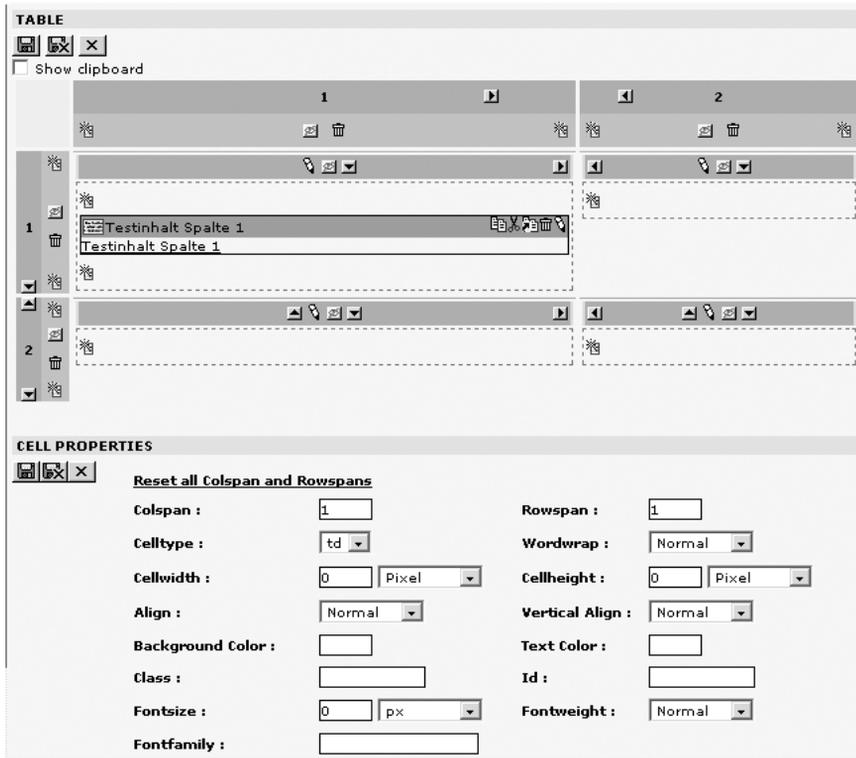


Abbildung 12.9 Konfigurationsoptionen für Tabellen mit KB Content Table

Accessible Tables

»Accessible Tables« erweitert das normale Inhaltselement »Tabelle« um einige Barrierefreiheitsfunktionen:

- ▶ `<thead>` `<tbody>` und `<tfoot>` zur Gruppierung von Tabellenbereichen
- ▶ `<caption>` zur Tabellenbeschreibung
- ▶ Tabellenkopfzeilen (`<th>`)
- ▶ Attribut `scop` in den Kopfzeilen
- ▶ Attribut `summary`

Außerdem gibt es die Möglichkeit, alle üblicherweise von TYPO3 angebrachten Klassen zu entfernen (bei einfachen Tabellen zu empfehlen) und `<p>` in den Tabellenzellen zu entfernen.

Diese Erweiterung ist eine gute Möglichkeit, einfache Tabellen barrierefrei zu gestalten – allerdings setzt sie auf die Verwendung von `scope` statt `headers`, was auch korrekt ist, aber in der Praxis kaum unterstützt wird.

2] Optionen für die Tabelle	
Beschreibung (<caption>)	<input type="text"/>
Zusammenfassung ("summary" Attribut im <table> tag)	<input type="text"/>
Fußzeile benutzen(<tfoot>)	<input type="checkbox"/>
Position der Kopfzeile	oben ▾
Alle CSS Angaben entfernen	<input type="checkbox"/>
CSS Klasse für diese Tabelle ("class")	<input type="text"/>

Abbildung 12.10 Zusätzliche Tabellenoptionen mit Accessible Tables

Sie können die Erweiterung unter dem Key *accessible_tables* finden – sie ist als »beta« eingestuft.

12.5.5 Dateiliste

Das Inhaltselement »Dateiliste« bietet verschiedene Layouts zur Darstellung von Dateilisten an – alle werden über Tabellen realisiert.

Nun kann man argumentieren, dass Dateilisten mit Dateinamen, Beschreibung, Icons und Dateigröße durchaus tabellarische Daten seien und daher die Verwendung einer Tabelle rechters wäre. Allerdings fehlen der resultierenden Tabelle all die oben besprochenen Barrierefreiheitseigenschaften, so dass sie in keinem Fall barrierefrei ist.

Wie genau eine Dateiliste zu realisieren ist, hängt auch von deren Komplexität ab:

Einfache Listen, die nur aus einem verlinkten Dateinamen bestehen, lassen sich hervorragend mit dem HTML-Element `` oder `` erstellen.

Auch das Element »Definitionsliste« (`<dl>`) eignet sich zur Darstellung einer Download-Liste – z. B. so:

```
<dl>
  <dt><a href="linkurl">Dateiname</a></dt><dd>Dateibeschreibung
    (PDF, 45kb)</dd>
</dl>
```

Für komplexe Listen schließlich ist eine Tabelle geeignet – dann allerdings mit Tabellenkopf und für Barrierefreiheit optimiert.

Der momentan einzige Weg dorthin führt über die Neu-Definition der Funktion **render_uploads**, die in **class.tx_cssstyledcontent_pi1.php** definiert ist (»CSS styled content«).

Dies kann z. B. über eine User-Funktion realisiert werden, die über TypoScript (**tt_content.uploads.20**) angesprochen wird.

12.5.6 Multimedia

Bei der Verwendung von Multimedia-Elementen ist sicher die Einbindung das kleinste Problem in Sachen Barrierefreiheit ...

Die BITV verlangt, zu Multimedia-Inhalten entsprechende Alternativen bereitzustellen. Bei Videos können dies Untertitel oder Audiodeskriptionen sein. Das W3C befürwortet den Einsatz der »Synchronized Multimedia Integration Language (SMIL)« – damit können Multimedia-Inhalte mit zusätzlichen Informationen, z.B. Untertiteln, ausgestattet werden.

Das Thema hier weiter zu vertiefen würde allerdings den Rahmen dieses Buches sprengen. Weitere Informationen zur Gestaltung von barrierefreien Multimedia-Inhalten finden Sie bei »Einfach für Alle« (<http://www.einfach-fuer-alle.de> (**Linkcode 054**) – füttern Sie die interne Suche mit »multimedia«) oder im Buch »Barrierefreies Webdesign« von Jan Eric Hellbusch.

Ein nützliches Software-Werkzeug zur Untertitelung von Multimedia-Daten (und mehr) ist MACpie (<http://ncam.wgbh.org/webaccess/magpie/> (**Linkcode 055**)).

TYPO3 setzt standardmäßig das Element `<embed>` für die Einbettung von Multimedia-Elementen ein. Leider ist `<embed>` kein offizielles HTML-Element, sondern eine proprietäre Erfindung von Netscape.

Zur Einbettung von Objekten sieht (X)HTML das Element `<object>` vor. Leider birgt die Einbettung mittels `<object>` einige Risiken (für Safari-Browser funktioniert sie leider nicht).

Geoff Stearns empfiehlt daher die Verwendung von Javascript zur Einbettung von Flash: <http://blog.deconcept.com/2005/03/31/proper-flash-embedding-flashobject-best-practices/> (**Linkcode 056**).

Für TYPO3 bedeutet dies, dass Multimedia-Objekte derzeit barrierefrei und standardkonform über das Inhaltselement »HTML« eingebunden werden (müssen).

12.5.7 Sitemap/Menü

Die Sitemap wird in der Standardeinstellung von TYPO3 ebenfalls über Tabellen realisiert. Es lässt sich jedoch ein barrierefreies Ergebnis über TypoScript-Änderungen bewerkstelligen. Hierfür müssen lediglich die diversen `<p>`-Tags, die um die einzelnen Navigationspunkte gewrappt werden, durch Listenauszeichnungen ersetzt werden.

Das TypoScript für eine Sitemap könnte dann folgendermaßen aussehen:

```
1: ### Barrierefreie Sitemap-Konfiguration ###
2: tt_content.menu.20.2 = HMENU
3: tt_content.menu.20.2.stdWrap.wrap =
4: tt_content.menu.20.2 {
5:     1 = TMENU
6:     1.noBlur = 1
7:     1.expAll = 1
8:     1.target = _self
9:     1.wrap = <ul class="sitemap">|</ul>
10:    1.NO.allWrap =
11:    1.NO.linkWrap =
12:    1.NO.allWrap = |<span class="usb">. </span>
13:    1.NO.wrapItemAndSub = <li class="sm1">|</li>
14:    1.NO.ATagBeforeWrap = 0
15:    2 < .1
16:    2.wrap = <ul>|</ul>
17:    2.NO.wrapItemAndSub = <li class="sm2">|</li>
18:    3 < .2
19:    3.NO.wrapItemAndSub = <li class="sm3">|</li>
20:    4 < .2
21:    4.NO.wrapItemAndSub = <li class="sm4">|</li>
22: }
```

Listing 12.2 Barrierefreie Sitemap durch TypoScript

Ausgegeben wird für das Sitemap-Menü eine verschachtelte ungeordnete Liste. Für die richtige Betonung beim Vorlesen wird ein `.` `` hinter jedem Menüpunkt gesetzt.

Dies ist notwendig, damit Screenreader beim Vorlesen nach jedem Menüpunkt die Stimme senken und eine kleine Pause machen und nicht alles in einem Rutsch »ohnepunktundkomma« vorlesen.

12.5.8 HTML

Das »HTML«-Element birgt von sich aus wenig Probleme – viel passiert dort ja auch nicht.

12.5.9 Mailformular

Formulare erfordern besondere Aufmerksamkeit, um keine unnötigen Barrieren aufzubauen. Dazu stehen dem Entwickler verschiedene Mittel zur Verfügung:

HTML-Element	Funktion
<fieldset>	Fast logisch zueinander gehörende Felder zusammen, z.B. die Felder »Name«, »Vorname« und »E-Mail« als persönliche Daten.
<legend>	Zu jedem Fieldset gehört eine Beschreibung der Gruppe (also eben »Persönliche Daten«).
<label>	Das Label kennzeichnet die Bezeichnung des Formularfeldes (beispielsweise »Name:«).

Tabelle 12.3 Bestandteile barrierefreier Formulare

Zusätzlich fordert die Barrierefreiheit, dass erforderliche Felder gekennzeichnet sind (üblicherweise durch ein *), dass aktive Felder markiert sind (per CSS sehr elegant möglich), und verlangt verständliche Fehlermeldungen bei der Validierung des Formulars. So sollte ein gutes Formular ohnehin aufgebaut sein.

Ferner muss die logische Struktur des Formulars durch die Reihenfolge der Felder gewahrt sein (nach dem Feld für den Vornamen sollte das Feld für den Nachnamen folgen). Wenn dies nicht durch die HTML-Struktur erreicht ist, kann dazu das Attribut `tabindex` eingesetzt werden. Es ist jedoch nicht unbedingt nötig, `tabindices` zu verwenden, wenn das Formular auch ohne diese logisch korrekt durchlaufen werden kann.

Prinzipiell nicht mehr nötig ist es, alle Felder generell vorzubelegen, wie es noch in der BITV gefordert ist. Diese Forderung ist von der technischen Entwicklung überholt. (In der BITV steht sie allerdings noch.)

Eine ausführliche Anleitung zur Erstellung barrierefreier Formulare hat »Einfach für Alle« zusammengestellt: <http://www.einfach-fuer-alle.de/artikel/formulare/> (**Linkcode 057**).

Um solche Formulare herzustellen, reicht das Inhaltselement »Mailform« nicht aus. Neben dem Tabellenlayout verhindern auch die fehlenden Elemente `<fieldset>`, `<legend>` und `<label>` den Einsatz.

Stattdessen kann die Erweiterung »th_mailformplus« (*th_mailformplus*) zur Erstellung barrierefreier Formulare verwendet werden. Hier kann ein HTML-Template zur Definition des Formulars eingesetzt werden.

Natürlich ist es nicht die ideale Lösung, auf HTML-Templates angewiesen zu sein – schließlich können die Redakteure nun keine eigenen Formulare mehr entwerfen. Wir haben daher in der schon angesprochenen Erweiterung »Accessible Content« den Formularwizard derart überarbeitet, dass er `<fieldset>`, `<legend>` und `<label>` verwendet und es somit auch dem Redakteur ermöglicht, einfache Formulare zu erstellen.

12.5.10 Suchformular

Das Standardformular für die Suche erzeugt ein per Tabelle positioniertes, nicht barrierefreies Formular. Es ist prinzipiell möglich, das Suchformular per TypoScript umzugestalten.

```
[mailform]=COA
[search]=COA
  [10]=lib.stdheader
  [20]=SEARCHRESULT
  [30]=FORM
    [layout]=<tr><td class="csc-form-labelcell">###LABEL###</td><td class="csc-form-fieldcell">###FIELD###</td></tr>
    [labelWrap]
    [commentWrap]
    [radioWrap]
    [REQ]=1
    [COMMENT]
    [target]=page
    [goodMess]=
    [badMess]=
    [locationData]=HTTP_POST_VARS
    [stdWrap]
    [wrap]=<table border="0" cellspacing="1" cellpadding="1" class="csc-searchform"> | </table>
    [editIcons]=tt_content:bodytext, pages, subheader
    [prefixComment]=2 | Search form inserted:
    [dataArray]
    [type]
    [no_cache]=1
```

Abbildung 12.11 TypoScript-Optionen der Suche

Das Erscheinungsbild der Suche kann über den TypoScript-Objektbrowser (siehe auch Abschnitt 9.5.3, *Template Tool: TypoScript Object Browser*) konfiguriert werden. Unter `tt_content.search.20` befinden sich die meisten Parameter, die das Erscheinungsbild des Formulars festlegen. Außerdem existiert eine TypoScript-Variablen `accessibility`. Wenn diese auf »1« gesetzt wird, werden alle `<label>` mit dem Attribut `for="..."` und der Bezeichnung des `input`-Feldes belegt.

Allerdings wird es ziemlich schwierig, tatsächlich auch `<fieldset>`-Elemente unterzubringen, da es keine offensichtliche Möglichkeit gibt, um Gruppen von Formularfeldern nebst zugehörigen `<label>` noch weitere Elemente zu »wrappen«.

Eine Alternative ist die Verwendung der – ohnehin wesentlich leistungsfähigeren – Extension »Indexed Search«. Dort lässt sich ein Formular als HTML einfügen und recht einfach konfigurieren (zusätzlich steht die Erweiterung »Searchbox for Indexed Search« (`macina_searchbox`) zur Verfügung, mit der ein eigenes HTML-Template für das Suchformular verwendet werden kann. Das neue Suchformular ist einfach als Inhaltselement auswählbar).

Leider ist die Ausgabe von »Indexed Search« nicht barrierefrei, so dass dann doch wieder der Blick in den Quellcode erforderlich ist. Der Aufbau der Ergebnisliste geschieht in der Datei `pi1/class.tx_indexedsearch.php` im Verzeichnis

der Extension. Wir zeigen in der Extension »Accessible Content« (siehe auch Kapitel 12.9, *Accessible Content*), wie eine solche umgestaltete Ergebnisseite aussehen kann.

12.5.11 Anmeldeformular

Auch das hauseigene Anmeldeformular führt nicht zu barrierefreiem Code. Und auch hier gibt es Alternativen – die Erweiterung »New front end login box« (*newloginbox*) bietet ohnehin mehr Möglichkeiten als das klassische Login. Darüber hinaus erlaubt es eine weitere Erweiterung »Loginbox with template for newloginbox plugin« (*arotea_loginbox*), eigene HTML-Templates einzusetzen, die barrierefrei gestaltet sein können.

12.5.12 Plugins

Bei den Plugins ist die Situation recht unterschiedlich. Einige sehr wichtige Extensions, wie z.B. »News« (*tt_news*) sind von Haus aus auf Barrierefreiheit getrimmt oder bieten die Möglichkeit, über Templates die Ausgabe zu beeinflussen.

Leider erzeugt die wichtige »Indexed Search« immer noch ein weitgehend unzugängliches Tabellengerüst – das soll sich erst mit TYPO3 4.0 ändern. Zumindest sollen dann Templates möglich sein.

12.5.13 Sprungmarken à la XHTML

Während in HTML 4 noch das Attribut »name« sowohl in Formularen als auch in benannten Anker () üblich war (und von TYPO3 standardmäßig eingesetzt wird), ist es in XHTML unerwünscht (>deprecated<) bzw. für FORM und IMG in der Variante »strict« und in XHTML 1.1 nicht mehr erlaubt.

Um standardkonforme Dokumente zu erstellen, müssen wir also sowohl die Anker selbst umbauen als auch dafür sorgen, dass automatisch erzeugte Links von TYPO3 entsprechend angepasst werden.

Da IDs auch nicht mit einer Zahl beginnen dürfen, reicht ein einfaches Umkopieren nicht aus. Stattdessen muss mit:

```
tt_content.stdWrap.dataWrap = <a name="anker{field:uid}"
id="anker{field:uid}"></a>|
```

noch ein Präfix vorangestellt werden. Für das Erzeugen der Links muss eine neue Funktion her (*userfunction*):

```
includeLibs.xhtmlanchor = fileadmin/user_ttypolink.inc
```

```
tt_content.text.20.parseFunc.tags.link.typolink.parameter.  
postUserFunc = user_xhtmlAnchor
```

Die zugehörige PHP-Funktion lautet (als **user_typolink.inc** in **fileadmin** speichern bzw. den Pfad oben entsprechend anpassen):

```
<?php  
function user_xhtmlAnchor($content,$conf) {  
    $uriparts = explode('#',$content);  
    if (isset($uriparts['1'])) {  
        $content = $uriparts['0'].'#anker'.$uriparts['1'];  
    }  
    return $content;  
}  
?>
```

Stellen, die automatisch Links auf Anker erzeugen, müssen auch angepasst werden. Das Menü **Abschnittsübersicht** (*sektion menu*) wird beispielsweise mit:

```
tt_content.menu.20.3.renderObj.typolink.section.wrap = anker|
```

angepasst. Diese Lösung stammt von Andreas Schwarzkopf und wurde in der TYPO3-Newsgruppe **typo3.projects.content-rendering** veröffentlicht.

12.6 Barrierefreie Menüs mit TYPO3

12.6.1 Textmenüs

Die beste Art, eine Navigation zugänglich zu machen, ist die Realisierung als Textmenü mittels des HTML-Elements ``.

Das ist zugänglich, semantisch korrekt (denn was ist ein Menü anderes als eine Liste von Links?) und lässt sich auch sehr gut per CSS stylen.

Wir gehen hier nicht weiter auf die Vorteile und Möglichkeiten von Listen als Menü ein – wenn Sie mehr zum Thema erfahren wollen, bieten diese Websites einen guten Einstieg:

- ▶ »Einfach für Alle«: Barrierefreie Navigationsmenüs (<http://www.einfach-fuer-alle.de/artikel/menues/tag1/> (Linkcode 058))
- ▶ »Listamatic« (Listen per CSS gestalten): <http://css.maxdesign.com.au/> (Linkcode 059)

Für die TypoScript-Realisierung hat eine Liste den Vorteil, dass sie recht einfach anzulegen ist. Im simpelsten Fall genügt Folgendes für ein dreifach verschachteltes Menü:

```
1: hmenu = HMENU
2: hmenu.1 = TMENU
3:   hmenu.1 {
4:     expAll = 0
5:     noBlur = 1
6:     wrap = <ul>|</ul>
7:     NO.wrapItemAndSub = <li>|</li>
8:     hmenu.2 = TMENU
9:     hmenu.2 < hmenu.1
10: hmenu.3 = TMENU
11: hmenu.3 < hmenu.2
```

Listing 12.3 TypoScript-Anweisungen für ein Menü als HTML-Liste

Um die Zugänglichkeit zu verbessern, ergänzen wir noch zwei Dinge:

1. eine versteckte Nummerierung für Screenreader
2. ».« zur Verbesserung der Aussprache

Gerade bei langen Menüs geht die Übersicht beim Vorlesen schnell verloren. Daher ist es ratsam, eine hierarchische Nummerierung vorzusehen (mehr dazu finden Sie auch im oben erwähnten »Einfach-für-Alle«-Artikel). Dazu nutzen wir das Definition-Element `<dfn>` und eine von Jan Wischnat geschriebene Benutzerfunktion.

Im TypoScript ergänzen wir:

```
hmenu.1.IProcFunc = user_IProc_dfn
```

Vorher müssen wir die User-Funktion einbinden:

```
page.includeLibs.dfn = fileadmin/inc/dfn_iproc_tmenu.inc
```

Sie finden die Funktion auf der beiliegenden CD-ROM.

Leider haben Screenreader die unangenehme Eigenschaft, aufeinander folgende Links direkt hintereinander vorzulesen, was die Übersicht nicht gerade verbessert. Daher müssen wir noch einen »Stopper« am Ende des Links einbauen, damit der Screenreader eine kleine Pause macht – wie am Ende eines Satzes.

```

<ul>
  <li><dfn>1:</dfn>
<a href="...">Link 1</a><span class="usb">. </span></li>
  ...
</ul>

```

Im TypoScript ist das einfach einzufügen:

```
NO.wrapItemAndSub = <li>|<span class="usb">. </span></li>
```

Da wir natürlich weder die Definition noch den Punkt sehen wollen, blenden wir sie per CSS aus und notieren im Stylesheet:

```

li dfn, .usb {
  display: block;
  position: absolute;
  left: -3000px;
  height: 0px;
  width: 0px;
}

```

Einfacher wäre `display:none`, aber auf diese Weise versteckte Elemente werden leider auch von den verbreitetsten Screenreadern nicht gelesen ...

Fertig ist das barrierefreie Textmenü – Sie können und sollten es nach Ihrem Bedarf mit zusätzlichen Optionen für TMENU ergänzen.

Ein Title-Attribut können Sie mit:

```
NO.ATagTitle.field = abstract // description
```

einem Menüstatus hinzufügen – hier dem normalen Zustand. Der Text für den TITLE wird dem Feld *abstract* entnommen – und wenn dieses leer ist, der *description* (die Verwendung des verlinkten Textes selbst als TITLE wäre allerdings eine sinnlose Verdoppelung und ist daher überflüssig).

Sinnvoll ist, die aktuelle Seite nicht zu verlinken.

Mit der Anweisung:

```

CUR < .NO
CUR.doNotLinkIt = 1

```

schalten Sie die Verlinkung für die aktuelle Seite ab.

Von der Verwendung von *Accesskeys* und *Tabindex-Markierungen* raten wir ab, da die meisten Buchstaben bereits durch andere Software vorbelegt sind und eigene Definitionen hier nur stören (lesen Sie dazu <http://2bweb.de/accesskey/> (Linkcode 060)). Für gut strukturierte Dokumente sind auch *Tabindizes* nicht erforderlich – bei der Verwendung von Sprungmenüs können sie sogar stören (auch hierzu gibt es eine weiterführende Quelle: <http://www.einfach-fuer-alle.de/artikel/tabindex/> (Linkcode 061)).

12.6.2 Grafische Menüs

Grafische Menüs der herkömmlichen Art – realisiert mit Tabellen und ohne Accessibility-Hilfen – stellen erhebliche Barrieren für behinderte Nutzer dar. Das fängt damit an, dass als Grafik realisierte Schrift nicht vergrößert werden kann. Wenn der Entwickler nicht wenigstens `alt`-Attribute vergeben hat, lesen Screenreader den Dateinamen vor, was meist wenig weiterhilft.

Daher ist aus Sicht der Barrierefreiheit generell von grafischen Menüs abzuraten – im Übrigen sind sie auch zur Suchmaschinenoptimierung und für die Download-Zeit eher schädlich.

Modernes Webdesign nutzt daher – wenn es denn Grafiken sein sollen – eine Technik, die »Image Replacement« genannt wird. Dabei wird ein normaler Text per CSS ausgeblendet und an seiner Stelle eine Hintergrundgrafik angezeigt. Es gibt inzwischen eine ganze Reihe verschiedener IR-Techniken (eine Diskussion der Vor- und Nachteile finden Sie unter <http://www.meiert.com/de/releases/20050513/> (Linkcode 062) von Jens Meiert).

12.7 Ein barrierefreier RTE

Für eine barrierefreie Ausgabe empfiehlt es sich, den RTE möglichst restriktiv zu konfigurieren. »Schädliche« Elemente wie `` können durch das Deaktivieren der Buttons »Textfarbe«, »Schriftart« und »Schriftgröße« unterbunden werden. Zusätzlich werden sie aus der Liste erlaubter Tags entfernt.

Wir haben der Konfiguration des RTE ein eigenes Kapitel gewidmet, so dass wir hier nur kurz auf die erforderlichen Maßnahmen eingehen – Details zur Konfiguration des RTE erfahren Sie in Abschnitt 7.3, *Rich Text Editor*.

Manche Funktionen sind mit dem Standard-RTE nicht realisierbar, so lassen sich `title`-Attribute erst mit der Version 0.5.1 des alternativen RTE »htmlArea RTE« zuweisen.

Ziel	Grund	Maßnahme
Keines der folgenden Elemente soll verwendet werden: <code></code> , <code></code> , <code><i></code> , <code><u></code>	Diese Elemente sind veraltet oder führen zu rein visuellem Code (z.B. sollte statt <code></code> <code></code> verwendet werden, das wird dann auch vom Screenreader entsprechend interpretiert).	Ausblenden der Buttons »Schriftart«, »Schriftgröße«, »Schriftfarbe«, »Emoticons«
Verwendung der TYPO3-Inhaltselemente, wenn möglich	Die Verwendung des Inhaltselements »Tabelle« gibt uns mehr Kontrolle über die Realisierung und verhindert doppelten Aufwand bei der Anpassung der Funktionen.	Ausblenden der Buttons zur Tabellendarstellung
Verwendung semantischer Elemente wie <code><cite></code> und <code><q></code> für Zitate	Semantisch korrekte Elemente können auch von Screenreadern und anderen Ausgabegeräten passend interpretiert werden.	Verwendung des Buttons »Benutzerdefinierte Elemente«
Verwendung der vorgegebenen CSS-Eigenschaften zur Gestaltung	Redakteure sollten keine eigenen Formatierungen einsetzen, sondern das verwenden, was vom Designer festgelegt wurde. Das ist nicht nur für die Barrierefreiheit sinnvoll.	Ausblenden des Buttons »Hintergrundfarbe«; verwenden von Klassen zur Gestaltung über die Buttons »Absatzart« und »Zeichenart« und Definition dieser Klassen im TypoScript.

Tabelle 12.4 Konfigurierung eines barrierefreien RTE

12.8 Einbindung von Tidy

Tidy ist ein sehr nützliches Werkzeug, um (X)HTML-Code zu prüfen und »aufzuräumen«. Ursprünglich von Dave Raggett entwickelt, wird das Programm inzwischen von einer Entwicklergemeinde als Sourceforge-Projekt gepflegt (<http://tidy.sourceforge.net/> (Linkcode 063)). Tidy existiert als Kommandozeilenversion für nahezu alle Betriebssysteme – es gibt sogar eine Online-Version (<http://www.thedumbterminal.co.uk/services/tidy.shtml> (Linkcode 064)).

Tidy prüft (X)HTML-Code auf Schreibfehler und falsch verschachtelte Tags, formatiert ihn mit Einrückungen übersichtlich und ersetzt gestalterische Anweisungen (``) durch entsprechende CSS-Eigenschaften. Auch zur Umwandlung von HTML in XHTML lässt sich Tidy verwenden.

Tidy ist als Kommandozeilen-Tool erhältlich und kann somit auch als nachgeordneter Parser für TYPO3-Code eingesetzt werden.

Tidy installieren und konfigurieren

Tidy liegt sowohl als kompilierte Version für viele Systeme als auch im Quellcode vor. Wie die Installation im Detail funktioniert, richtet sich (natürlich) nach Ihrem Betriebssystem und ist in der Regel unkompliziert. Selbst für Windows liegt eine einfache ausführbare Datei vor, die keine spezielle Installation erfordert.

TYPO3 sieht bereits eine Verwendung von Tidy vor – im Install-Tool können Sie im Bereich »All Configuration – FE« (bzw. in `$TYPO3_CONF_VARS["FE"]`) Tidy einschalten (`[FE][tidy] = 1`) und konfigurieren (`[FE][tidy_option] = ..`). Außerdem müssen Sie dort den Pfad auf dem Server angeben (`[FE][tidy_path] = ...`).

Die im Install-Tool vorbelegten Parameter haben folgende Bedeutung:

-i	Rückt den Quellcode ein.
-quiet true	Es werden keine Meldungen durch Tidy ausgegeben.
-tidy-mark true	Es wird ein Metaelement eingesetzt, das auf die Reinigung durch Tidy hinweist.
-wrap 0	Tidy bricht zu lange Zeilen nach einer hier anzugeben Anzahl Zeichen um. Mit »0« wird dies abgeschaltet.
-raw	Gibt Zeichen oberhalb von 127 aus ohne sie in Entities umzuwandeln.

Tabelle 12.5 Parameter im Install-Tool

Eine detaillierte Liste aller Tidy-Optionen finden Sie auf der CD-ROM oder unter <http://tidy.sourceforge.net/docs/quickref.html> (**Linkcode 065**). Wenn Sie Tidy installiert haben, können Sie mit `tidy -h` und `tidy -help-config` alle verfügbaren Konfigurationsparameter einsehen.

12.9 Accessible Content

Die vorhandenen Rendering-Funktionen von TYPO3 sind (noch) nicht für barrierefreie Ausgabe optimiert. Obwohl prinzipiell eine barrierefreier Quellcode möglich ist, bedeutet es doch an vielen Stellen erhebliche Mühe und Detailarbeit. Wir haben daher eine Extension geschrieben, die uns und Ihnen diese Arbeit erleichtern soll.

»Accessible Content« bietet neue Rendering-Funktionen für die meistverwendeten Inhaltstypen und Content-Objekt-Typen. Durch den Einsatz dieser Extension und des mitgelieferten statischen Templates ist es möglich, bestehende TYPO3-Projekte zu barrierefreien Webseiten umzubauen und

neue, barrierefreie Webseiten aufzubauen. Jedoch ist diese Extension dabei nur eine Unterstützung – Barrierefreiheit lässt sich nicht auf rein technische Hilfen reduzieren!

»Accessible Content« (Key: *sb_accessiblecontent*) ist von der Systemextension »CSS Styled Content« abgewandelt und soll diese für barrierefreie Webseiten ersetzen. Sie stellt ein eigenes *static template* zur Verfügung.

12.9.1 Exkurs: Rendering von Content-Elementen

Das Rendering von Inhaltselementen wird im wesentlichen von der Klasse `tslib_cObj` vorgenommen. Der Administrator hat über das TypoScript-Template umfangreiche Möglichkeiten, die Frontendausgabe zu verändern. Diese Möglichkeiten sind jedoch begrenzt. Soll beispielsweise der Inhaltstyp »Text mit Bild« statt mit Tabellen CSS-basiert ausgegeben werden, so ist dies nicht über das Template einstellbar.

Inhaltstelemente werden in der Tabelle `tt_content` abgelegt. Die Spalte `CType` gibt an, um welchen Inhaltstyp es sich handelt. Die eigentliche Zusammenstellung der Typen findet im TypoScript-Setup unter `tt_content` statt. Quelle dieser Definition ist meist das verwendete *static template*.

Ein Inhaltstyp wird aus mindestens einem Content-Objekt, in den meisten Fällen einem COA (*content object array*), zusammengesetzt. Nachfolgend ist dies am Beispiel des Typs `textpic` aus dem *static template CSS Styled Content* beschrieben:

```
1: tt_content.textpic = COA
2: tt_content.textpic {
3:     10 = COA
4:     10.if.value = 25
5:     10.if.isLessThan.field = imageorient
6:     10.10 = < lib.stdheader
7:     20 = < tt_content.image.20
8:     20.text.10 = COA
9:     20.text.10 {
10:         if.value = 24
11:         if.isGreaterThan.field = imageorient
12:         10 = < lib.stdheader
13:         10.stdWrap.dataWrap = <div class="csc-textpicHeader
           csc-textpicHeader-{field:imageorient}">|</div>
14:     }
```

```

15:     20.text.20 = < tt_content.text.20
16: }

```

Listing 12.4 Rendering eines Elementes »Bild mit Text« (textpic)

Die Funktionsweise im einzelnen:

- ▶ In `tt_content.textpic.10` und `tt_content.textpic.20.10` wird das Feld `imageorient` aus `tt_content` abgefragt. Je nachdem, ob es kleiner oder größer gleich 25 ist, wird der Standard-Header in den einen oder anderen Container gelegt. Dieses Feld wird in der Eingabemaske für das Inhaltselement im Feld »Position« gesetzt. Die Verknüpfung zwischen dem Zahlenwert und der wählbaren Position ist im `$TCA` eingestellt.
- ▶ In `tt_content.20` wird das Bild abgelegt, es wird dafür der Eintrag aus `tt_content.image.20` kopiert.
- ▶ Im Bereich `text` von `tt_content.20` wird dann der Text abgelegt, auch hier handelt es sich um eine Kopie eines Teils eines anderen Inhaltstypen, nämlich `text`. In diesem Inhaltstypen wird auf den Content-Objekt-Typen `TEXT` zurückgegriffen.

Für jedes Content Objekt gibt es in der Klasse `tslib_cObj` eine eigene Rendering-Funktion. Im Falle von `HRULER` sieht die Funktion beispielsweise so aus:

```

1: function HRULER ($conf) {
2:     $lineThickness = tslib_div::intInRange($this->stdWrap
3:         ($conf['lineThickness'],$conf['lineThickness'],1,50);
4:     $lineColor=$conf['lineColor']?$conf['lineColor']:'black';
5:     $spaceBefore = intval($conf['spaceLeft']);
6:     $spaceAfter = intval($conf['spaceRight']);
7:     $tableWidth = $conf['tableWidth'] ? $conf['tableWidth'] :
8:         '99%';
9:     $content='';
10:    $content.='<table border="0" cellspacing="0" cellpadding="0"
11:        width="'.htmlspecialchars($tableWidth).'"><tr>';
12:    if ($spaceBefore) {$content.='<td width="1"></td>'; }
15:    $content.='<td bgcolor="'. $lineColor.'"></td>';

```

```

11:   if ($spaceAfter) {$content.='<td width="1"></td>'; }
12:   $content.='</tr></table>';
13:   $content = $this->stdWrap($content, $conf['stdWrap.']);
14: return $content;
15: }

```

Listing 12.5 Rendering-Funktion für HRULER

In `$conf` werden das TypoScript für das aktuelle Element übergeben, hier sind es `width`, `height`, `wrap` und `stdWrap`. Ohne auf die Details dieser Funktion einzugehen, so ist doch klar zu erkennen, dass sie nicht – wie man es von einem HRULER, also einer horizontalen Linie, erwarten würde, mit einem `<hr>`-Tag arbeitet, sondern stattdessen auf Tabellenkonstrukte zurückgreift.

12.9.2 Neues Rendering erstellen

Im vorangegangenen Abschnitt wurde das Rendering einzelner Inhaltselemente erklärt. Das Beispiel der Funktion `HRULER()` macht deutlich, dass viele der HTML-Codes, die in der Ausgabe erscheinen, hartcodiert in der Klasse `tslib_cObj` vorhanden und daher über TypoScript nicht verändert werden können.

Deshalb ist es notwendig, diese Funktionen durch eigene auszutauschen. Das kann auf zwei Wegen erfolgen:

- ▶ Über eine XCLASS von `tslib_cObj`. Leider der zurzeit einzige Weg, die Rendering-Funktionen direkt auszutauschen, da ein Hook fehlt. (Zu XCLASS finden Sie mehr Informationen im Abschnitt 13.10.4).
- ▶ Über die Umgestaltung des TypoScript-Setups für Inhaltstypen – die Standard-Rendering-Funktionen werden nicht verwendet, stattdessen werden User-Funktionen (Content-Objekt-Typ `USER`) verwendet (dazu mehr im Abschnitt 13.6.4).

In »Accessible Content« werden vorerst beide Varianten benutzt. Dies hat historische Gründe: da »Accessible Content« aus »CSS Styled Content« hervorgegangen ist und dort einige Inhaltstypen durch User-Funktionen ausgetauscht wurden, sollten diese vorerst beibehalten werden. Für unsererseits hinzugefügte Rendering-Funktionen verwendeten wir aber lieber das XCLASS-Konzept, da wir direkt die Content-Objekte ändern wollten.

Letztendlich ist eine direkte Änderung der Content-Objekte (und damit die erstgenannte Variante) der von uns bevorzugte Weg, da das Konzept der Con-

tent-Objekte es auf einfachem und übersichtlichem Wege erlaubt, Frontend-Ausgaben baukastenartig zusammenzubauen, statt für jeden Inhaltstypen User-Funktionen zu schreiben. Zudem greifen viele Extensions auf die Content-Objekte zurück, so dass eine Änderung der Inhaltselement-Ausgabe nicht ausreichen könnte, um eine barrierefreie Webseite zu erstellen.

Zukünftig wird »Accessible Content« den Weg der XCLASS weiterverfolgen. Wo es notwendig ist, werden Content-Objekte ausgetauscht. Sollte es in späteren Varianten einen Hook geben, der es möglich machen, direkte Austauschfunktionen für die Content-Objekt-Renderingfunktionen anzubieten, so wird die Extension entsprechend umgestellt.

12.9.3 Die XCLASS

»Accessible Content« ändert in der XCLASS `ux_tslib_cObj` die in der Tabelle genannten Content Objekte.

Content Objekt	Änderungen
FORM	<p><code><textarea></code> und <code><input></code> erhalten einen Default-Inhalt, der per JavaScript weggenommen wird, wenn der Besucher das Feld anklickt.</p> <p>Es wird ein neuer Feldtyp <code><fieldset></code> definiert.</p>

Tabelle 12.6 Änderungen an Content-Objekten in der XCLASS `ux_tslib_cObj`

Wie »CSS Styled Content« auch, stellt »Accessible Content« eine Pluginklasse mit diversen User-Funktionen bereit. Diese User-Funktionen dienen dem von den Content Objekten entkoppelten, speziell auf das Inhaltselement zugeschnittenen, Rendering.

Inhaltstyp	User-Funktion
textpic	<code>tx_sbaccessiblecontent_pi1->render_textpic()</code>
bullets	<code>tx_sbaccessiblecontent_pi1->render_bullets()</code>
table	<code>tx_sbaccessiblecontent_pi1->render_table()</code>
uploads	<code>tx_sbaccessiblecontent_pi1->render_uploads()</code>

Tabelle 12.7 User-Funktionen der Plugin-Klasse in »Accessible Content«

Alle Methoden geben barrierefreien Quelltext aus.

12.9.4 Tabellen

Eine barrierefreie Tabelle benötigt eine Zusammenfassung (*Summary*), eine Tabellenüberschrift (*Caption*) sowie Kopfzeilen (<th> statt <td>), auf die sich die sonstigen Zellen beziehen. Nachfolgend ist der HTML-Quelltext einer barrierefreien Tabelle schematisch dargestellt:

```
1: <table summary="Zusammenfassung">
2: <caption>Dies ist eine Beispieltabelle</caption>
3:   <tr>
4:     <th scope="col" id="col0">Überschrift a</th>
5:     <th scope="col" id="col1">Überschrift b</th>
6:   </tr>
7:   <tr>
8:     <td headers="col0">Inhalt 1</td>
9:     <td headers="col1">Inhalt 2</td>
10:  </tr>
11: </table>
```

Listing 12.6 Barrierefreie Tabelle

Folgende Unterschiede bestehen zu einer normalen mit TYPO3 erstellbaren Tabelle:

- ▶ Der <table>-Tag erhält ein Attribut `summary`, in dem eine kurze Zusammenfassung des Tabelleninhaltes steht.
- ▶ Unterhalb des <table>-Tags und überhalb der ersten Zeile erscheint ein <caption>-Tag, der die Tabellenüberschrift beinhaltet
- ▶ Es gibt Kopfzeilen, die mittels <th>-Tag gesetzt werden. Ob sie sich auf eine Spalte oder eine Zeile definieren, wird in ihrem Attribut `scope` definiert. Sie erhalten ausserdem ein `id`-Attribut.
- ▶ Alle Inhaltzellen, die den <td>-Tag, erhalten ein `headers`-Attribut, das sich auf die `id` des <th>-Tags bezieht.

Die Standardversion des Table Wizard von TYPO3, die in Abbildung 12.13 dargestellt ist, enthält all diese Möglichkeiten nicht. Daher wurde für »Accessible Content« ein eigener, von der Standardvariante abgeleiteter Editor geschrieben (Abbildung 12.14). Ist die Extension installiert, so erhält man beim Klick auf das Wizard Icon den neuen *Table Wizard*.

Legt man ein Inhaltselement vom Typ »Tabelle« an, so erhält man nach Eingabe der Überschrift und erstem Speichern das Wizard Icon, das einen Link auf den *Table Wizard* enthält. Dies verdeutlicht Abbildung 12.12.



Abbildung 12.12 Inhalt der Tabelle und Wizard Icon

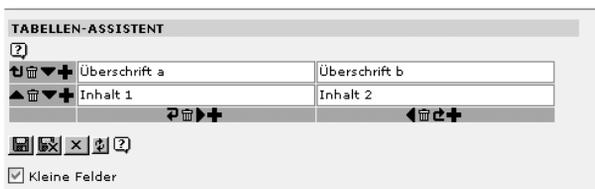


Abbildung 12.13 Der Table Wizard von TYPO3

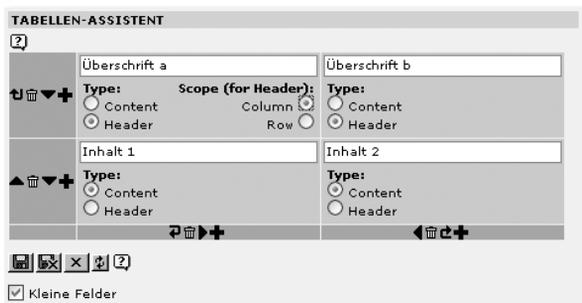


Abbildung 12.14 Ein neuer Table Wizard von Accessible Content

Wie Abbildung 12.12 zeigt, werden die Tabelleninhalte im Textformat gespeichert. Dabei ist | das Trennzeichen zwischen den Spalten und ein Zeilenumbruch das Trennzeichen zwischen den Zeilen.

Der neue *Table Wizard* hat die Möglichkeit, zusätzliche Zelleninformationen zu speichern. Es kann zwischen »Content« (<td>) und »Header« (<th>) unterscheiden und bei solchen Tabellenköpfen, wo die Zuordnung nicht eindeutig ist, der Scope bestimmt werden. Diese Zusatzinformationen werden mittels eines neuen Trennzeichens, der Tilde ~, abgespeichert (siehe Abbildung 12.15):



Abbildung 12.15 Neues Speicherformat der barrierefreien Tabellen

Jede Zelle trägt nun die zusätzlichen Informationen. Um diese auch ausgabeseitig zu berücksichtigen, wurde eine User-Funktion für das Rendering geschrieben. Im TypoScript wird dafür das Rendering für das entsprechende Content Object geändert (gekürzt):

```
tt_content.table = COA
tt_content.table {
    10 = < lib.stdheader
    20 = USER
    20.userFunc = tx_sbaccessiblecontent_pil->render_table
    20.innerStdWrap.parseFunc = < lib.parseFunc
}
```

Somit ist es nun möglich, barrierefreie Tabellen zu erstellen. Genauere Informationen darüber, wie ein Wizard ausgetauscht werden kann, finden Sie im Abschnitt 13.4.4, *\$TLA-Betrieb: Wizard austauschen*.

12.9.5 Formulare

Die Umgestaltung der Formulare gestaltete sich ähnlich wie diejenige der Tabellen. Auch hier wurde der Wizard ausgetauscht, um zusätzliche Informationen eingeben zu können. Eingabeseitig war hier im Wesentlichen dafür Sorge zu tragen, dass es möglich ist, das Formular in mittels `<fieldset>` gesetzte Sektionen zu unterteilen. Dazu wurde im Form Wizard ein neuer Elementtyp »Fieldset« angelegt. Ein gesetztes Fieldset wird entweder am Ende des Formulars oder beim Öffnen eines neuen Fieldsets geschlossen.

Abbildung 12.16 zeigt einen beispielhaften Eintrag im neuen Formular-Wizard. Der resultierende Quelltext ist nachfolgend dargestellt:

```
1: <form action="index.php?id=6" name="8e6"
    enctype="multipart/form-data" method="post" target="page"
    onsubmit="return validateForm('8e68dd22fa267c12b607c6602d5e3b
    1c','name,Name%3A,email,e-Mail','','')"><input
    type="hidden" name="locationData" value="6:tt_content:12" />
2:     <fieldset>
3:         <legend>Kontaktdaten</legend>
4:         <label for="name">Name:</label>
5:         <input type="text" name="name" id="name" size="40"
            onfocus="if(this.value=='Ihr Name')this.value='';"
            onblur="if(this.value=='')this.value='Ihr Name';"
            value="Ihr Name" />
6:         <label for="email">e-Mail</label>
```

```

7:      <input type="text" name="email" id="email" size="40"
        onfocus="if(this.value=='Ihre Mailadresse')this.
        value='';" onblur="if(this.value=='')this.value=
        'Ihre Mailadresse';" value="Ihre Mailadresse" />
8:    </fieldset>
9:  </fieldset>
10: <legend>Eintrag</legend>
11: <label for="comment">Ihr Kommentar</label>
12: <textarea name="comment" id="comment" cols="30" rows="10"
    wrap="virtual" onfocus="if(this.value=='Kommentar')this.
    value='';" onblur="if(this.value=='')this.value=
    'Kommentar';">Kommentar</textarea>
13: </fieldset>
14: </fieldset>
15: <legend>Absenden</legend>
16: <label for="formtype_mail"></label>
17: <input type="submit" name="formtype_mail"
    id="formtype_mail" value="Abschicken" />
18: </fieldset>
19: </form>

```

Listing 12.7 Formular über den verbesserten Formular-Wizard

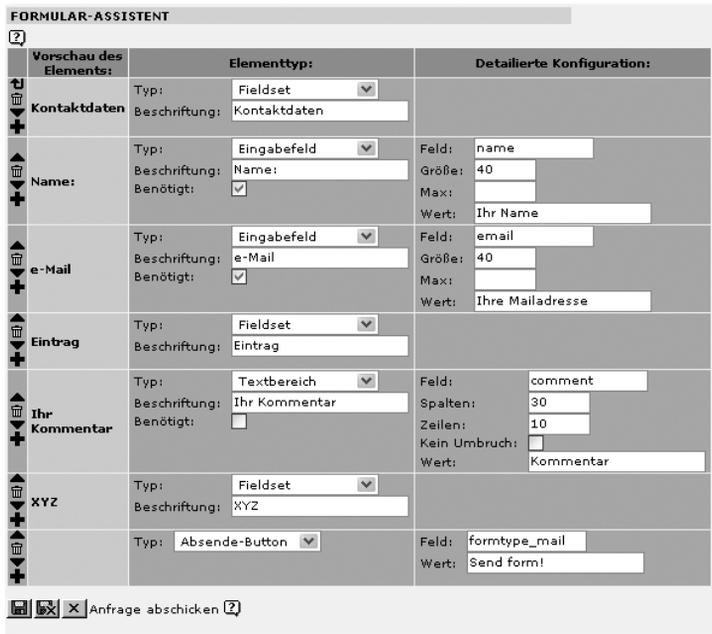


Abbildung 12.16 Eingabe eines Formulareintrags im neuen Formular-Wizard

Wie zu sehen, ist das Formular entsprechend der Vorgaben in `<fieldset>`-Sektionen unterteilt. Die Beschriftungen aller Eingabefelder sind in einem `<label>`-Tag verpackt, der in seinem `for`-Attribut angibt, auf welches Formularfeld er sich bezieht. Zusätzlich sind textbasierte Eingabefelder bereits mit einem Eingabewert vorgefüllt. Beim Anklicken dieser Eingabefelder sorgt ein JavaScript dafür, dass der Inhalt gelöscht wird. Verlässt man ein Eingabefeld und hat nichts eingetragen, so trägt das JavaScript den Vorgabewert wieder ein.

Anders als bei den Tabellen wurde die für Formulare zuständige Methode `FORM()` in der XCLASS umgeschrieben. Zukünftig ist es geplant, stattdessen auf eine User-Funktion zurückzugreifen.

12.9.6 Probleme mit Plugins

Im Gegensatz zu »CSS Styled Content« arbeitet »Accessible Content« zur Zeit als reine Extension, für die keine expliziten Core-Änderungen vorliegen. Das führt zu einem Problem mit der Frontendausgabe von Plugins, für das wir vorerst keine elegante Lösung anbieten können.

Fügt man ein Plugin, beispielsweise die »New front end login box« (`newloginbox`) ein, so erscheint dieses in der Frontendausgabe nicht. Verantwortlich dafür ist der in vielen Extensions verwendete Befehl `t3lib_extMgm::addPItoST43()`, der nur für die *Static Templates* »Content (default)« und »CSS Styled Content« funktioniert.

Abhilfe schafft hier leider nur, die von diesem Befehl getätigten Eintragungen im TypoScript für jede einzelne Extension vorzunehmen; hier ein Beispiel für die »New front end login box«:

```
tt_content.list.20.newloginbox_pi1 = < plugin.tx_newloginbox_pi1
tt_content.list.20.newloginbox_pi3 = < plugin.tx_newloginbox_pi3
```

Die entsprechenden Eintragungen können aus den Extension-Dateien **ext_tables.php** oder **ext_localconf.php** abgelesen werden. Diese sehen für die `newloginbox` wie folgt aus:

```
t3lib_extMgm::addPItoST43($_EXTKEY, 'pi1/class.tx_newloginbox_pi1.php', '_pi1', 'list_type', 0);
t3lib_extMgm::addPItoST43($_EXTKEY, 'pi3/class.tx_newloginbox_pi3.php', '_pi3', 'list_type', 0);
```

Wichtig sind der erste, zweite und dritte Parameter, die sich in TypoScript wie folgt wiederfinden:

```
tt_content.list.20.[Parameter 1][Parameter 3] = <
plugin.[Parameter 2 (nur Klassenname)]
```

Für einige Plugins haben wir die Einträge bereits im *Static Template* vorgenommen.

Sonderfall Indexed Search

Die Frontend-Ausgabe der beliebten Extension »Indexed Search Engine« (*indexed_search*) ist alles andere als barrierefrei. Daher haben wir auch für diese Extension neue Ausgabefunktionen geschrieben. Leider ist bei »Indexed Search Engine« die komplette Ausgabe direkt in der Plugin-Klasse realisiert – es stehen keine Templates wie z.B. für »News« zur Verfügung. In der aktuellen Version von »Accessible Content« haben wir daher Anpassungen in einer Klassendatei **class.ux_tx_indexedsearch.php** untergebracht, die in `ext_localconf.php` die Original-Klasse ersetzt.

In der mit TYPO3 3.8 mitgelieferten Version der Indexsuche wurden einige Funktionen verändert, so dass unsere Klasse leider nicht mehr funktioniert. Wir werden auf der Buchseite und unter **<http://typo3.sunbeam-berlin.de/accessiblecontent/>** (Linkcode 066) eine neue Version unserer Klassenbibliothek bereitstellen.

So lange können Sie – wenn Sie »Accessible Content« auf einer 3.8er-Version einsetzen – die Modifikation von »Indexed Search Engine« deaktivieren, indem Sie in der Datei **ext_localconf.php** (im Verzeichnis der Extension »Accessible Content«) die folgende Zeile auskommentieren:

```
$TYPO3_CONF_VARS[TYPO3_MODE] ["XCLASS"] ['ext/indexed_search/  
pi/class.tx_indexedsearch.php'] = t3lib_extMgm::extPath($_EXT-  
KEY)."class.ux_tx_indexedsearch.php";
```

12.9.7 Weitere Entwicklung von »Accessible Content«

Die Arbeit an der Extension ist weit davon entfernt, abgeschlossen zu sein – vielleicht haben ja Sie auch den ein oder anderen Hinweis?

Wir werden die Arbeit an der Extension fortsetzen und die aktuelle Entwicklung auf der Website zum Buch und unter folgendem Link dokumentieren:

<http://typo3.sunbeam-berlin.de/accessiblecontent/> (Linkcode 066)

Index

\$cObj->dataArray 315
\$TCA 128, 462, 481, 484
\$TYPO3_CONF_VARS 625, 628,
630, 631, 635
* 297, 298
.special 326
»HELLO WORLD!«-Ausgabe 290

A

Abkürzungen 426
Absatzarten 134, 136
Access List 205
Accessibility 199, 438
Accessible Content 445, 446, 493
Accessible Tables 426, 433
Admin 525
Administratorpasswort 66
AdminPanel 98
Akronym 426
Akronymmanager 426, 518
All Configuration 625
Allgemeine Datenschutzverwaltung
95
Allgemeine Seitenkonfiguration
316
Allgemeinen Datensatzsammlung
363
Allgemeiner Datensatz → Allge-
meine Datenschutzverwaltung
Allow excludfields 207
alt-Attribut 421
Alternative Editoren 128, 141
altLabels 121
API 595
Arbeitsbereich 509
Aufgabenlisten 226
Ausgabekonfiguration 139
Ausgangspunkt 506

B

Backend → Benutzeroberfläche
Backend Benutzer 96
Backend Benutzergruppen 96
Backend-Benutzer 192, 202, 215, 524
Backend-Module 509
Backup 107
Barrierefrei 424
Barrierefreie Formulare 201
Barrierefreie Sitemap 436
Barrierefreier RTE 443
Barrierefreiheit 419
BASE 210
Basis-Template 303
Basis-Templates 299, 316
Bedingungen 305
Benutzerklassen 219
Benutzeroberfläche 81
Benutzerrechte 223
Benutzerverwaltung 192, 226
BIENE-Wettbewerb 421
bigDoc 510
BITV 419, 421
Breadcrumb-Navigation 326
Browser 81
Browsereinstellungen 81
Brute-Force-Angriff 67
Bulletlist 425

C

Caching 463, 526, 607, 623
Caching → Clear all cache
Caption 450
CGLcompliance 466
charset 318
cHash 549, 610
Checkbox 483
Chefredakteure 203
Clear all cache 313

- CMS 482
- COA 532
- COA, COA_INT → cObject
- cObject
 - CASE 274
 - CLEAR GIF 282
 - COBJ_ARRAY 269
 - COLUMNS 283
 - CONTENT 272
 - CTABLE 283
 - EDITPANEL 282
 - FILE 270
 - FORM 277
 - HMENU 273
 - HRULER 284
 - HTML 268
 - IMAGE 270
 - IMG_RESOURCE 271
 - IMGTEXT 284
 - LOAD_REGISTER 276
 - MULTIMEDIA 281
 - OTABLE 284
 - PHP_SCRIPT 280
 - RECORDS 273
 - RESTORE_REGISTER 276
 - SEARCHRESULT 279
 - TEMPLATE 281
 - TEXT 268
 - USER und USER_INT 279
- cObjects 267
- columns 483
- Comments 307
- Conditions 262
- Conditions → Bedingungen
- Constant Editor
 - cat 308
 - label 310
 - type 310
- Constants 265
- Constants display 305
- constants.txt 314
- Content (default) 425, 508
- Content Object Array 532
- Content Objects → cObjects
- Content Rendering → Rendering
- Context Sensitive Help 494
- Cross Site Scripting 598
- CSH 494
- CSS 546
- CSS Styled Content 424, 508
- CSS Styled Content → Standard-Templates
- CSS Styled Imagetext 425
- CSS-Menü 639
- ctrl 482
- CType 501

D

- DAL 476
- Database Abstraction Layer 476
- Database Analyzer 622
- Datei-Browser 575
- Dateifreigaben 97
- Dateiliste 425
- Dateistruktur 462, 464
- Datenbanktabellen 478
- Datensatz kopieren 499
- Datensatz löschen 499
- Datensatz verschieben 499
- Datentypen 260
- DB Mounts 209
- DBAL 476, 540
- Definitionsliste 434
- Deleted 485
- die() 64
- doctyp 317
- Doctype 422
- doctype 317
- Doctype-Deklarationen 317
- Dokumentation 604
- Dummy Package 63

E

- Edit-Panel 554
- Eigenschaften 251
- embed 435
- Encryption Key 67, 628
- Exclude-field 487
- Explicitly allow/deny field values 202
- Explicitly allow/deny values 208
- Extended Table Backend 431
- Extension
 - csh_de 82
 - CSS Styled Content 385
 - Kickstarter 359
 - Native Workflow System 228
 - News 360
 - User>Task Center, Tasks 226
 - User>Taskcenter, Quicknote 226
 - User>Taskcenter, Recent 226
 - User>Taskcenter, Root Records 226
 - Versioning Management 233
 - Workflow 233
- Extension Development Evaluator 588, 602
- Extension Key 349, 461
- Extension-Manager 463
- Extensions 343
 - Content (default) 131
 - CSS Styled Content 131
 - Full Backup 108
 - htmlArea RTE 144
 - Pages Drag'n Drop 90
- extList 631

F

- FCKeditor 146
- Fehlern bei der Klammersetzung 305
- felInterface 482
- Feldbeschreibung 95

- Feldnamen der Bild-Inhaltsobjekte 127

- Feldnamen der Text-Inhaltsobjekte 125, 127

- Fieldset 449, 452
- File Mounts 209
- Filemount → Dateifreigaben
- Flash-Plugin 420
- Flexform 483, 558
- Formulare 452
- FreeType 620, 627
- Frontend-Benutzer 192
- Frontend-Benutzerverwaltung 192
- Frontend-Editing 553
- Frontend-Rendering 507, 526
- Funktionen 261
- Funktionsmenü 522

G

- GD-Bibliothek 620, 624
- GDLib 620
- Geschützter Bereich 192, 194, 220
- GET 515, 536, 548, 597
- Globale Extensions 344, 464
- Globale Variablen 473
- GMENU 322
- GMENU_FOLDOUT 322
- GMENU_LAYERS 322
- Gov Textmenu 426
- gov_accessibility 425
- gov_accesskey 426
- grafische Menüs 323
- GraphicsMagick 620

H

- Hauptframe 523
- Hauptmodul 83, 509
- Header 609
- Hello World 559
- Hidden 485
- HMENU 322
- Hook 449

Hooks 592
HRULER 448
HTMLarea 142
htmlArea RTE 443
HTML-Vorlagen 543
Hyperlinks 547

I

Image Processing 623
ImageMagick 620
IMGMENU 322
Import/Export → T3D-Dateien
includeLibs 532
Indexed Search 337, 438
Inhaltselement 504
Inhaltstyp 501, 531
Inhaltstypen 91
 Bild 93
 Dateilinks 93
 Datensatz einfügen 94
 Formular 94
 HTML 95
 Login 94
 Menü/Sitemap 94
 Multimedia 94
 Plugin einfügen 94
 Punktliste 93
 Skript 94
 Suchen 94
 Tabelle 93
 Textbox 94
 Trenner 94
 Überschrift 93
Input-Feld 483, 568
Installation 63
 Installationstool 64
Interface 482

J

JavaDoc 604
JavaScript 546
JSMENU 322

K

Karteikartenmenü 515
Kaskadierung 294
KB Content Table 432
Kern 68
Kern → TYPO3-Kern
Kickstarter 485, 518, 599
Kommentare 254
Kommentare → Comments
Konfiguration 66
Konstanten → Constants

L

lang-Attribute 427
language key 317
Layout 506
LDAP 72
 ActiveDirectoryServices 72
 eDirectory 72
 OpenLDAP 72
Lightweight Directory Access
 Protocol 72
LINK 129
Link 547
localconf.php 621
Locallang-Datei 495
Locallang-XML-Datei 584
Lock to Domain 204
lockIP 633, 638
lockSSL 633
Login 81, 194
Login-Formular 194, 334
Logische Strukturen 420
lokale Extensions 345, 464
Löschen und Leeren von
 Konstanten 294

M

Mailformular 436
Main Module 509
maxFileSize 634
MD5 67

Mediafeld 337
mediumDoc 510
mehrdimensionale PHP-Arrays 257
Mehrsprachigkeit 584
Menüzustände
 ACT 322
 AVTIFSUB 322
 CUR 322
 IFSUB 322
 NO 322
 RO 322
 SPC 322
 USR 323
Meta tags, extended 320
Meta-Daten 586
Metainformationen 481
Metatags 319
MM-Relation 478
Modul
 Benutzer 85
 Benutzer>Aufgaben 230
 Benutzer>Einstellungen 82
 Datei 85
 Datei>Bilder 210
 Datei>Dateiliste 210
 Dokumente 85
 Hilfe 86
 Tools 86
 Tools>Benutzer Administrator
 223
 Tools>Ext Manager 343, 347
 Tools>Workflow 234
 Web 84
 Web>Dashboard 234, 239
 Web>Liste 95, 204
 Web>Template 196
 Web>Zugriff 202, 219
Module 83, 509
Modules 206
Modulleiste 509, 523
Mount 524

Multimedia 435
MySQL 475
N
Namenskonventionen 461
Navigationsbereich 523
Navigationsframe 523
New front end login box 439
News → RSS
News-Ausgabe 365
Newsfeed 382
noDoc 510
numerischer Objekte 292
numerisches Array 252

O
object 435
Objekte 251
Objektnamen 251
Objekttypen 251
Operatoren 254
Optimierung 607
Optionsplit 327

P
page record → Seitendatensatz
Page TSconfig 111, 314
 mod 112
 RTE 113
 TCEFORM 115
 TCEMAIN 114
 TSFE 115
Page types 207
Page Validator 423
Page-Browser 575
pagegen 527
pageTSconfig.txt 133
Palette 484
palettes 484
Parent ID 479
php.ini 619

PHP-Array → numerischer Objekte
 PHP-Arrays 243
 PHP-Objekte 243
 PHP-Version 465
 piVars 539
 Plugin 504
 Plugin-Klasse 506
 Plugins 546
 Popup-Fenster 549
 POST 536, 548, 597
 Primary Key 479, 499
 Project Coding Guidelines 466
 Proxy 609
 Punktnotation 257

Q

Quickstart Package 63
 Quotierung 477

R

Radiobuttons 483, 576
 Rechtevererbung 202
 Redakteure 203
 Reiter 558, 566
 relationale Datenbankstruktur 493
 removeDefaultJS 318
 removeItems 121
 Rendering 244, 314, 446, 528
 Rendering-Objekt 315
 Rich Text Editor 128
 Rollen 202
 Root 95, 290, 300
 Rootline 290, 300
 Rootline-Arrays 300
 Root-Seite 203, 226
 Root-Template 290, 303
 RSS 381
 RSS → Newsfeed
 RTE 81, 483, 570
 RTE_imageStorageDir 631
 RTE-API → RTE-Schnittstelle
 RTEenabled 631
 RTE-Konfiguration 113, 120, 132
 RTE-Schnittstelle 128
 RTE-Transformationen 128
 RTE-Transformationen → Transformationsmodi

S

safe_mode 619
 Schreibweisen 253
 Seite 96
 Seite verstecken 88
 Seiten anlegen
 Anlegen neuer Seiten 87
 Seitenbrowser 510
 Seitendatensatz 87
 Seiteninhalte 91
 Seiten-Navigator 575
 Seitentyp
 Abstand 89
 Backend Benutzer Bereich 89
 Erweitert 88
 Externe URL 89
 Mount Seite 89
 Nicht im Menü 89
 Papierkorb 90
 Shortcut 89
 Standard 88
 SysOrdner 90
 Seitenzugriff 219
 Seitenzugriffsrechte → Zugriffsrechte
 Sektion 513
 Select-Feld 483, 573
 sendmail 619
 Services 595
 Session-Management 534
 setMemoryLimit 630
 setup.txt 314
 Sheets 566
 Sitemap 333, 435
 smallDoc 510
 Sprungmarken 439

- SQL 477
- SQL-Injections 598
- SSL 633
- Standard-Rich-Text-Editor → RTE
- Standard-Template 303
- Standard-Templates 296
 - aus Extensions 303
 - content(default) 298
 - cSet 298
 - css_styled_content 298
 - frameset 298
 - language.* 298
 - plugin.* 297
 - plugin.alt* 297
 - styles.* 297
 - template 297
- Static Template 507, 589
- stdWrap 261, 534
- Sub Group 202
- Sub Modules 509
- Submodul 509
- Submodule 84
- Submodules 516
- Suchformular 438
- Summary 450
- Symlink 526
- Syntax 251
- Syntax Highlighting → Syntax-HL
- Syntax-HL 307
- System-Extensions 344, 464

T

- T3D-Dateien 104
- Tab Menu 515
- Tabelle 429, 450
- Table 425
- Table Configuration Array 481
- Table Configuration Array → \$TCA
- Table Wizard 450
- Tables (listing) 207
- Tables (modify) 202, 207
- TCE 481, 498
- TCEforms 481, 498
- template record → Template-Datensatz
- Template Tool
 - Constant Editor 294, 305
 - CSS Styler 313
 - Info/Modify 301
 - Template Analyzer 247, 303
 - TypoScript Object Browser 303
- Template-Datensatz 301
- Template-Kaskadierung → Kaskadierung
- Templates on next Level 300
- Template-Tool
 - Info/Modify 291
- Testsite Package 63
- Text m/Bild 93
- Text w/summary 501
- Textarea 483, 569
- Text-Bild 428
- Textmenü 324, 440
- th_mailformplus 437
- Tidy 444
- TinyMCE 147
- TLO → Top-Level-Objekte
- TMENU 322
- TMENU_LAYERS 322
- To-Do-Listen → Aufgabenlisten
- Toolbar 132
- Top-Level-Objekte 286
 - config 287
 - constants 287
 - FEData 287
 - includeLibs 287
 - lib 288
 - PAGE 286
 - plugin 288
 - resources 287
 - sitetitle 287
 - styles 288

- temp 288
- tt_* 288
- types 287
- Transformationsfunktionen 130
- Transformationsmodi 131
- Transformationswege 129
- TScnfig 109, 314
- TSOB → TypoScript Object
 - Browser
- TTFdpi 627
- types 483
- TYPO3 Core Engine 481, 498
- TYPO3-Kern 68
- TYPO3-Online-Repository 356
- TYPO3-Tags 131
- typolink 611
- TYPOLIST 129
- TypoScript 243
- TypoScript Object Browser 291
- TypoScript-Constants 293
- TypoScript-Kommentare →
 - Kommentare
- TypoScript-Menüs 322
- TypoScript-Referenz 247
- TypoScript-Schreibweisen →
 - Schreibweisen
- TypoScript-Setup 291, 507
- TypoScript-Syntax → Syntax
- TypoScript-Template 481
- TypoScript-Templates 290
- TypoScript-Wizard 110

U

- Überschrift
 - Text 93
- Unterobjekte 252

- URL 549
- User TScnfig 115, 314
 - admPanel 116
 - mod 116
 - options 116
 - setup 117
- User-Funktion 507
- UTF-8 585

V

- Validator 423
- Verschieben von Seiten 90
- Versionssprünge 358

W

- W3C 421
- Website Sprache 97
- Wertzuweisung 251
- Wizard 493
- Workflows 224

X

- XCLASS 590
- XHTML 421
- XHTML-Cleaning 317, 422
- XML 475, 558, 584
- XML-Prolog 317, 422
- XSS 598

Z

- Zeichenarten 134, 137
- Zugriffsrechte 202, 219, 525
- Zugriffsrechte → Benutzerklassen
- Zugriffsrechte → Seitenzugriff
- Zweite Optionspalette 95