

FACHHOCHSCHULE FLENSBURG

Fachbereich Angewandte Informatik

**BACHELORTHESIS**

**im Studiengang Medieninformatik**

**Thema:** Integration der Datenbank-Abstraktionsschicht Doctrine2  
in das Content-Management-System TYPO3

**eingereicht von:** Stefan Kowalke <stefan.kowalke@stud.fh-flensburg.de>

**Matrikelnummer:** 485366

**Abgabedatum:** 25. April 2014

**Erstprüfer:** Prof. Dr. Hans-Werner Lang

**Zweitprüfer:** Dipl. VK Tobias Hiep

Dieses Dokument wurde am 25. April 2014 mit  $\LaTeX 2_{\epsilon}$  gesetzt.

Schrift: 12pt-TODO  
Typographie: 2012/07/29 v3.11b KOMA-Script  
System: Lua $\TeX$ -0.76.0 auf OSX 10.8  
Editor: Mou 0.8.5 beta und VIM 7.4

Dieses Werk steht unter der **Creative Commons Namensnennung – Weitergabe unter gleichen Bedingungen 4.0 International Lizenz**. Es kann unter <https://github.com/Konafets/thesis> heruntergeladen werden.



Der für diese Thesis entstandene Prototyp steht unter der GNU General Public License version 2 oder neuer.

## ABSTRACT

---

In der Defaulteinstellung speichert das CMS TYPO3 die Inhalte der Website in einer MySQL-Datenbank. Soll sich das CMS mit der Datenbank eines anderen Herstellers verbinden, muss die mitgelieferte Systemextension *DBAL* installiert werden. Damit die Anfrage von der *DBAL* Extension geparkt und in die anderen SQL-Dialekte konvertiert werden kann, müssen sich die Entwickler - sei es TYPO3 Kern- oder externer Extensionprogrammierer - an gewisse Richtlinien, wie zum Beispiel die ausschließliche Nutzung der TYPO3 Datenbank API, halten. Trotz der Nutzung der API können zu *DBAL* inkompatible Anfragen formuliert werden. Ferner ist es möglich komplett an der API vorbei mit der Datenbank zu kommunizieren, wodurch man wieder beim Anfangsproblem angelangt ist, welches man mit der Nutzung der *DBAL* Extension lösen wollte: Die Fixierung auf einen SQL-Dialekt. Desweiteren verwendet die *DBAL* Extension die externe Bibliothek *AdoDB* welche nicht mehr aktiv weiterentwickelt wird und weder Verbesserungen noch Sicherheitsupdates bereitstellt. Ziel dieser Arbeit war es zum einen die veraltete Bibliothek gegen eine andere - in dem Fall Doctrine 2 *DBAL* - zu ersetzen, die in sich unter aktiver Entwicklung befindet, und zum anderen einen einheitlichen Zugriff auf die Datenbank zu erlauben, der die direkte Kommunikation mit der Datenbank verbietet. Es wurde eine Extension geschrieben, die die TYPO3 eigene Datenbank API überschreibt und anstelle dessen eine eigene API anbietet, die über Doctrine *DBAL* und PDO auf die Datenbank zugreift. Dabei wurde auf die Abwärtskompatibilität zur TYPO3 eigenen API geachtet, dass noch nicht angepasster Code weiterhin mit der Datenbank kommunizieren kann. Zu Test- und Demonstrationszwecken wurde der Kern von TYPO3 an die neue API angepasst. Dies erlaubt das manuelle Testen der Funktionsweise des Backends und des Frontends - es wurden jedoch auch Unit Tests für die API implementiert, wodurch das auch das automatische Testen ermöglicht wird.



“‘The Answer to the Great Question... Of Life,  
the Universe and Everything... Is... Forty-two,’ said Deep  
Thought, with infinite majesty and calm.”

– Douglas Adams, The Hitchhiker’s Guide to the Galaxy

## INHALTSVERZEICHNIS

---

<b>Abstract</b>	<b>iii</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Grundlagen</b>	<b>5</b>
2.1 TYPO3 CMS . . . . .	5
2.1.1 Geschichte . . . . .	5
2.1.2 Definition . . . . .	6
2.1.3 Architektur und Aufbau von TYPO3 CMS . . .	7
2.1.4 Extensions . . . . .	12
2.2 Doctrine 2 DBAL . . . . .	17
2.2.1 Beschreibung . . . . .	17
2.2.2 Abgrenzung zum ORM . . . . .	17
2.2.3 Verbreitung . . . . .	17
2.3 PDO . . . . .	17
2.4 Versionskontroll- und Issuetrackingsystem . . . . .	17
2.5 Unit Testing . . . . .	17
2.6 Sicherheit . . . . .	17
<b>3 Prototypischer Nachweis der Herstellbarkeit</b>	<b>18</b>
3.1 Refactoring der alten Datenbank API . . . . .	18
3.1.1 Tests für die alte Datenbank API . . . . .	18
3.2 Testgetriebene Implementierung der neuen Datenbank API . . . . .	18
3.2.1 Einführung von Query Objekten . . . . .	18
3.3 Anwendung der neuen Datenbank API . . . . .	18
3.4 Überprüfen der Funktionalität . . . . .	18
<b>4 Ausblick</b>	<b>19</b>
<b>Zusammenfassung</b>	<b>20</b>
<b>Abkürzungsverzeichnis</b>	<b>22</b>
<b>Eidesstattliche Erklärung</b>	<b>26</b>

## EINLEITUNG

---

Als sich auf den Developer Days 2006 das Entwicklerteam für einen Nachfolger der eben erst erschienenen TYPO3 Version 4.0 formierte (vgl. [TYP08]), war wohl keinem der dort Anwesenden klar wohin die Reise gehen würde – ging man anfänglich noch von einem Refactoring<sup>1</sup> der schon vorhandenen Codebasis aus.

In der Konzeptionsphase kristallisierte sich immer mehr heraus, dass es damit nicht getan sein würde. Der Nachfolger mit dem Arbeitstitel “Phoenix” sollte nicht nur den zukünftigen Anforderungen des Web standhalten, sondern die Position der Version 4.0 weiter ausbauen. Das Entwicklerteam um Cheftwickler Robert Lemke entschloss sich die Version 5.0 des Systems komplett neu zu schreiben [Quelle anfügen] und merkte dabei, dass Entwickler bei der Programmierung von Webanwendungen immer wieder mit den gleichen Problemen wie Routing, die Erstellung und Validierung von Formularen, Login von Benutzern oder dem Aufbau einer Verbindung zur Datenbank konfrontiert werden.

Die Idee eines – von dem Content-Management-System – unabhängigen PHP Frameworks war geboren und wurde zunächst auf den Namen FLOW3 getauft. Dieses Framework sollte die spätere Basis für TYPO3 5.0 bilden und all die oben beispielhaft angeführten wiederkehrenden Aufgaben übernehmen. Die Version 5.0 von TYPO3 sollte lediglich eins von vielen Packages darstellen mit denen FLOW3 erweitert werden kann. Vielmehr wurde es als eigenständiges “Webapplication Framework” konzipiert und umgesetzt, so dass es auch ohne ein Content-Management-Sytem (CMS) betrieben werden kann und auch wird. [Quelle zu Rossmann einfügen].

Schon in einer recht frühen Entwicklungsphase hat man sich dem Thema Persistenz gewidmet, die zunächst noch als “Content Repository for Java Technology API (JCR)” in PHP implementiert, jedoch später wegen zu vieler Probleme bei der Portierung der Java Spezifikation JSR-170 nach PHP durch eine eigene Persistenzschicht ersetzt wurde (vgl. [Dam10]). Im weiteren Verlauf der Entwicklung kam man von dieser Idee wieder ab, da die eigene Persistenzschicht nicht performant genug war und andere Projekte wie Doctrine oder Propel schon fertige Lösungen anboten (vgl. [DEM14]). Schlußendlich entschied man

---

<sup>1</sup> Strukturverbesserung des Quellcodes bei Beibehaltung der Funktionalität  
Version: Commit: 00ad2e7 from 2014-04-24 21:08:28 +0200

sich für die Integration von Doctrine als Persistenzschicht, da der Hauptentwickler von Doctrine, Benjamin Eberlei, seine Hilfe anbot.

Für die Anwender stellt sich bei einem Versionssprung stets die Frage, ob eine Migration von der alten zur neuen Version möglich ist und mit wieviel Aufwand dies verbunden sein würde. Diesen Bedenken folgend trafen sich die Kernentwickler beider Teams 2008 in Berlin, um die Routemaps beider Projekte in Einklang zu bringen. Als ein Ergebnis dieses Treffens wurde das "Berlin Manifesto" (vgl. [TYP08]) bekanntgegeben, welches mit klaren Worten feststellt<sup>2</sup>:

“

- TYPO3 v4 continues to be actively developed
- v4 development will continue after the the release of v5
- Future releases of v4 will see its features converge with those in TYPO3 v5
- TYPO3 v5 will be the successor to TYPO3 v4
- Migration of content from TYPO3 v4 to TYPO3 v5 will be easily possible
- TYPO3 v5 will introduce many new concepts and ideas. Learning never stops and we'll help with adequate resources to ensure a smooth transition

*Die TYPO3 Kernentwickler*

An der Umsetzung wurde sofort nach dem Treffen begonnen, indem Teile des FLOW3 Frameworks nach TYPO3 Version 4.0 zurück portiert und unter dem Namen *Extbase* als Extension veröffentlicht wurden. Es erfüllt zu gleichen Teilen die Punkte 3 und 6 des Manifests, da es die neuen Konzepte aus FLOW3 der Version 4.0 zur Verfügung stellt und somit gleichzeitig diese Version näher an die Technologie des Frameworks heranzführt.

Die Aufgabe von Extbase besteht darin ein Application Programming Interface (API) bereitzustellen, mit denen Entwickler von Extensions auf die internen Ressourcen und Funktionen von TYPO3 CMS zugreifen und das System somit nach eigenen Wünschen und Anforderungen erweitern können, ohne den Code des CMS selbst verändern zu müssen. Es ist als vollständiger Ersatz der bis dahin angebotenen PI-

<sup>2</sup> Mittlerweile wird TYPO3 Neos innerhalb der Community nicht mehr als der Nachfolger von TYPO3 CMS angesehen. Es stellt lediglich – wie TYPO3 Flow – ein weiteres Produkt innerhalb der TYPO3 Familie dar. [Quellenangabe]



Base API [LINK ZU PI BASE] konzipiert worden, wobei es aktuell noch möglich ist sich für einen der beiden Ansätze zu entscheiden.

Extbase führt per Definition einige – bis dahin in TYPO3 v4 unbekannte – Programmierparadigmen ein. Als größter Unterschied zu dem PI-Based Ansatz ist hier sicherlich das Model-View-Controller (MVC) Pattern zu nennen. Dabei werden die Daten im Model vorgehalten, der View gibt die Daten aus und der Controller steuert die Ausgabe der Daten. Das Model ist unabhängig von der View, was bedeutet, dass die gleichen Daten auf verschiedene Weise ausgegeben werden können. Man denke hier an Meßdaten, die zum einen als Tabelle über einer Listview dargestellt werden können oder als Diagramme mit einer entsprechenden View.

Das Model – eine herkömmliche PHP Klasse – wird dabei von Extbase automatisch auf die Datenbank abgebildet, so dass ein Objekt eine Zeile darstellt und dessen Eigenschaften als Spalten der Tabelle interpretiert werden. Diese Technik wird als Objektrelationale Abbildung (engl. Object-relational mapping (ORM)) genannt. Das zum Einsatz kommende ORM ist Bestandteil der oben erwähnten selbstgeschriebenen Persistenzschicht von FLOW3, da Extbase zu der Zeit rückportiert wurde, als diese bei FLOW3 im Einsatz war.

Obwohl Extbase beständig weiterentwickelt wird und es der Wunsch der Community ist, die in darin verwendete Persistenzschicht gegen Doctrine 2 auszutauschen, was sich in Form von Posts auf der Mailingliste (vgl. [TYP13]) oder in Prototypen ausdrückt (vgl. [Mar12] und [Ebe12]), ist dies bis heute noch nicht realisiert worden. Der Chefentwickler von Doctrine, Benjamin Eberlei, hat gegenüber dem Autor in einer persönlichen Korrespondenz die unterschiedlichen Ansätze beider Projekte wie folgt zum Ausdruck gebracht:

“ (...) Doctrine nutzt das Collection interface, Extbase SplObjectStorage. Doctrine Assoziationen funktionieren semantisch anders als in Extbase, z.B. Inverse/Owning Side Requirements. Typo3 hat die Enabled/Deleted flags an m\_n Tabellen, sowie das start\_date Konzept. Das gibts in Doctrine ORM alles evtl nur über Filter API, aber vermutlich nicht vollständig abbildbar. Das betrifft aber alles nur das ORM, das Doctrine Database Abstraction Layer (DBAL) hinter Extbase zu setzen ist ein ganz anderes Abstraktionslevel.

*Benjamin Eberlei per E-Mail vom 17.12.13 00:12*

Zum jetzigen Zeitpunkt wird die DBAL in TYPO3 durch eine Systemextension [Glossareintrag] bereitstellt, die auf der externen Bibliothek AdoDB basiert, welche jedoch Anzeichen des Stillstands aufzeigt und davon ausgegangen werden kann, dass das Projekt nicht weiterentwickelt wird. [Linkt zu SourceForge]

Anhand dieser Fakten wird ersichtlich, dass die Integration von Doctrine erstrebenswert ist, da dadurch die Abhängigkeit zu dem inaktiven Projekt AdoDB aufgelöst werden kann. Da jedoch eine Integration von Doctrine ORM in Extbase nicht in der gegebenen Zeit, die für die Bearbeitung der Thesis zur Verfügung steht, zu realisieren ist, wurde der Fokus stattdessen auf die Integration von Doctrine CMS in TYPO3 gelegt, wodurch nicht nur Extbase von den Möglichkeiten eines DBAL profitieren kann, sondern der gesamte Core und somit alle Extensions die noch nicht mit Extbase erstellt worden sind.

Ferner wird durch diesen Ansatz eine stabile Basis zu Verfügung gestellt, auf der eine zukünftige Integration der ORM Komponente von Doctrine in Extbase aufbauen kann.

Ziel dieser Thesis ist es einen funktionierenden Prototypen zu entwickeln, der zum einen aus einer Extension besteht, die für die Integration von Doctrine DBAL zuständig ist und zum anderen aus einem modifizierten TYPO3, welches die neue API, die mit der Extension eingeführt wird, beispielhaft benutzt.

Im ersten Teil werden die eingesetzten Werkzeuge vorgestellt. Es wird erklärt warum diese und nicht andere eingesetzt worden sind und wie diese in Hinblick auf die Aufgabenstellung benutzt wurden.

Der zweite Teil beschreibt die praktische Umsetzung und schließt mit einer Demonstration wie der Prototyp getestet werden kann.

Teil drei gibt einen Ausblick auf die weitere Verwendung des Quellcodes und des Prototypen, während Teil vier mit einem Fazit schließt.

Im Anhang befindet sich neben den obligatorischen Verzeichnissen für Literatur und Abbildungen ein Glossar, sowie das Abkürzungsverzeichnis.

## GRUNDLAGEN

---

### 2.1 TYPO3 CMS

#### 2.1.1 Geschichte

TYPO3 CMS ist ein Web Content Management-System (WCMS) und wurde von dem dänischen Programmierer Kaspar Skårhøj im Jahr 1997 zunächst für seine Kunden entwickelt - im Jahr 2000 von ihm unter der GNU General Public License v.2 (GPL2) veröffentlicht. Dadurch fand es weltweit Beachtung und erreichte eine breite Öffentlichkeit. Laut der Website T3Census<sup>1</sup> gab es am 7. April 2014 208561 Installationen von TYPO3 CMS.

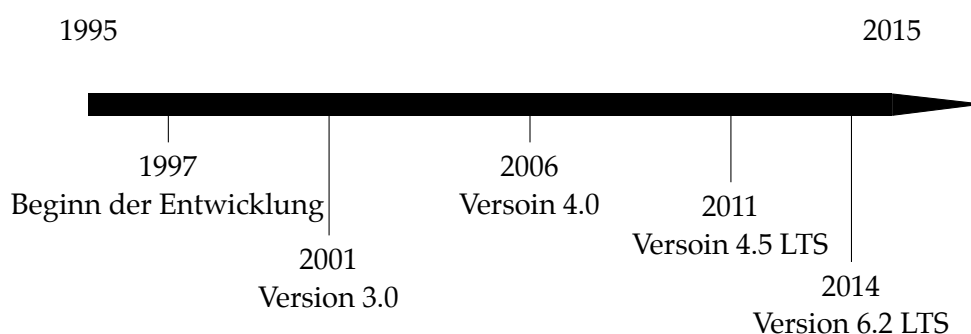


Abbildung 2.1: Zeitachse der TYPO3 CMS Entwicklung

Im Jahr 2012 entschied sich das Projekt zu einer Änderung in der Namensgebung:

- aus TYPO3 v4<sup>2</sup> wurde TYPO3 CMS
- aus FLOW3 wurde TYPO3 Flow
- und aus TYPO3 5.0 / TYPO3 Phoenix wurde TYPO3 Neos

---

<sup>1</sup> <http://t3census.info/>

<sup>2</sup> Damit ist das von Skårhøj entwickelte CMS gemeint, welches den 4.x Zweig des Projekts darstellt.

Diese Änderung wurde notwendig, da schon länger abzusehen war, dass TYPO3 Phoenix nicht den Nachfolger von TYPO3 v4 darstellt. Somit war die Entwicklung von TYPO3 v4 in dem Versionszweig 4.x gefangen und es konnten keine neuen Features eingebaut oder veraltete Funktionen entfernt werden. Durch dieses neue Schema bekommt der Name "TYPO3" die Bedeutung einer Dachmarke zuteil, während "TYPO3 CMS", "TYPO3 Flow" und "TYPO3 Neos" Produkte innerhalb der TYPO3 Familie darstellen. Im weiteren Verlauf dieser Arbeit werden ausschließlich die neuen Namen verwendet.

Heute kümmert sich ein Team um die Entwicklung von TYPO3 CMS und eines um TYPO3 Flow und TYPO3 Neos. Dahinter steht keine Firma, wie es bei anderen Open Source Projekten wie Drupal (Acquia) oder Wordpress (Automattic) vorzufinden ist, sondern die TYPO3 Association (T3Assoc). Die T3Assoc ist ein gemeinnütziger Verein und wurde 2004 von Kaspar Skårhøj und anderen Entwicklern gegründet um als Anlaufstelle für Spenden zu dienen, die die langfristige Entwicklung von TYPO3 sicherstellen sollen. Die Spenden werden in Form von Mitgliedsbeiträgen erhoben.<sup>3</sup>

### 2.1.2 Definition

TYPO3 CMS ist ein klassisches CMS, welches auf die Erstellung, die Bearbeitung und das Publizieren von Inhalten im Intra- oder Internet spezialisiert ist und es somit per Definition zu einem WCMS macht.

Daneben findet man auch die Bezeichnung Enterprise Content Management-System (ECMS)<sup>4</sup>, was als Hinweis auf den Einsatz des Systems für mittel- bis große Webprojekte dient.

Als letztes sei noch erwähnt, dass TYPO3 CMS ebenso zu den Content Management Framework (CMF) gezählt werden kann, da es dem Entwickler verschiedene APIs zur Verfügung stellt. Dieser Begriff findet sich unter anderen in einem Kommentar im – von TYPO3 CMS erzeugten – HTML-Code:

```
<!--
  This website is powered by TYPO3 - inspiring people to share!
  TYPO3 is a free open source Content Management Framework
  initially created by Kasper Skaarhoj and licensed under GNU/GPL.
  TYPO3 is copyright 1998-2012 of Kasper Skaarhoj. Extensions
  are copyright of their respective owners.
  Information and contribution at http://typo3.org/
-->
```

<sup>3</sup> <http://association.typo3.org/>

<sup>4</sup> <http://www.typo3.org>

### 2.1.3 Architektur und Aufbau von TYPO3 CMS

Im folgenden werden die grundlegenden Konzepte von TYPO3 CMS vorgestellt. Dort wo es für das weitere Verständnis notwendig ist, wird tiefer in das Thema eingestiegen. Ansonsten werden die Konzepte lediglich angerissen um einen generellen Überblick zu erhalten.

#### *Webstack als Basis*

TYPO3 CMS wurde in PHP<sup>5</sup> - basierend auf dem Konzept der Objekt-orientierung - geschrieben und ist damit auf jeder Plattform lauffähig, die über einem PHP Hypertext Processor Interpreter verfügt. Die Version 6.2 von TYPO3 CMS benötigt mindestens PHP Hypertext Processor 5.3.7.

PHP Hypertext Processor bildet zusammen mit einem Apache Webserver und einer MySQL Datenbank den sogenannten Webstack, der abhängig von dem eingesetzten Betriebssystem MAMP (OSX / Mac), LAMP (Linux) oder WAMP (Windows) heißt.

In der Standardeinstellung kommt MySQL als Datenbank zum Einsatz - durch die Systemextension [Glossar] können jedoch auch Datenbanken anderer Hersteller angesprochen werden. Eine genaue Analyse dieser Extension erfolgt im Kapitel [KglsapITEL zur Analyse von ext:DBAL einfügen].

#### *Ansichtssache*

Aus Anwendersicht teilt sich TYPO3 CMS in zwei Bereiche:

- das Backend  
stellt die Administrationsoberfläche dar. Hier erstellen und verändern Redakteure die Inhalte während Administratoren das System von hier aus konfigurieren
- das Frontend  
stellt die Website dar, die ein Besucher zu Gesicht bekommt.

(vgl. [DRB08, S. 5])

---

<sup>5</sup> PHP steht für "PHP: Hypertext Preprocessor"

## Der Systemkern und die APIs

TYPO3 CMS besteht aus einem Systemkern, der lediglich grundlegende Funktionen zur Datenbank-, Datei- und Benutzerverwaltung zu Verfügung stellt. Dieser Kern ist nicht monolithisch aufgebaut, sondern besteht aus Systemextensions. (vgl. [Lab+06, S. 32])

Die Gesamtheit aller von TYPO3 CMS zur Verfügung gestellten APIs, wird als die "TYPO3 API" bezeichnet. Diese kann - analog zum Backend / Frontend Konzept - in eine Backend API und eine Frontend API unterteilt werden kann. Die Aufgabe der Frontend API ist die Zusammenführung der getrennt vorliegenden Bestandteile (Inhalt, Struktur und Layout) aus der Datenbank oder dem Cache zu einer HTML-Seite. Die Backend API stellt Funktionen zur Erstellung und Bearbeitung von Inhalten zur Verfügung. (vgl. [DRB08, S. 5 ff.])

Die APIs, die keiner der beiden Kategorien zugeordnet werden kann, bezeichnet [DRB08, S. 5 ff.] als "Common"-API. Die Funktionen der Common-API werden von allen anderen APIs genutzt. Ein Beispiel dafür stellt die Datenbank API dar, welche in der Regel nur einfache Funktionen wie das Erstellen, Einfügen, Aktualisieren, Löschen und Leeren<sup>6</sup> von Datensätzen bereitzustellen hat. Würde man je eine Datenbank API für das Frontend und das Backend zur Verfügung stellen, bricht man eine wichtige Regel der Objekt-orientierten Programmierung - Don't repeat yourself. Dieser - mit hoher Wahrscheinlichkeit - redundanter Code würde die Wartbarkeit des Programms verschlechtern und die Fehleranfälligkeit erhöhen.

Auf die aktuelle Datenbank-API wird in [KAPITEL zur Analyse der aktuellen Situation einfügen] näher eingegangen.

## Verzeichnisstruktur

Im Gegensatz zu früheren TYPO3 CMS Versionen gibt es kein "Dummy"-Package<sup>7</sup> mehr. Ab Version 6.2 enthält der Download lediglich den TYPO3 CMS Kern in Form des Verzeichnisses *typo3/*.

Dieses Verzeichnis ist außerhalb des Webroots abzulegen. Im Webroot ist ein Verzeichnis *www.example.com* anzulegen, in dem die Verzeichnisse *fileadmin/*, *typo3conf/*, *typo3temp/* und *uploads/* anzulegen sind. Das Verzeichnis *typo3\_src/* ist ein (Linux) Symlink auf das Installationsverzeichnis von TYPO3 CMS und das Verzeichnis *typo3/* ist ebenfalls ein Symlink, welcher auf über den Symlink *typo3\_src* auf *typo3* zeigt. Dieser Aufbau macht ein Update recht einfach, da lediglich der Symlink

<sup>6</sup> CRUD - Create, Retrieve, Update und Delete

<sup>7</sup> Damit ist ein weitgehend leeres Paket gemeint, dass alle Dateien enthält die im Webroot des Servers laufen sollen. Es stellt einen Container für die spätere Website dar.

```

.
├── Packages/
│   └── Libraries/
├── fileadmin/
├── typo3_src/ -> ../../typo3-6.2.0
├── typo3/ -> typo3_src/typo3
│   ├── contrib/
│   ├── ext/
│   ├── gfx/
│   ├── install/
│   ├── js/
│   ├── mod/
│   └── sysext/
├── typo3conf/
│   ├── ext/
│   │   ├── doctrine_dbal/
│   │   └── phpunit/
│   └── l10n/
├── typo3temp/
├── uploads/
├── media/
├── pics/
├── tf/
└── tx_phpunit/

```

Abbildung 2.2: Verzeichnisstruktur von TYPO3 CMS

*typo3\_src* auf das Installationverzeichnis der neuen Version "umgebo-gen" werden muss.

Im folgenden werden die einzelnen Verzeichnisse näher erklärt:

Im Verzeichnis *www.example.com* muss noch ein Symlink *index.php* an-gelegt werden, welcher auf *typo3\_src/index.php* zeigt.

Unter *www.example.com/typo3conf/* befindet sich die Datei *LocalConfiguration.php*. Diese enthält die Grundkonfiguration in Form eines Arrays. Darin sind verschiedenen Einstellungen festgelegt:

- Debug Mode
- Sicherheitslevel für den Login (Fronend und Backend)
- das Passwort für das Installtool (mit MD5 und Salt ghasht)
- die Zugangsdaten zur Datenbank (Benutzername, Password, Da-tenbankname, Socket, ...)
- Einstellungen zum Caching

Verzeichnis	Erklärung
Packages/Libraries/	Dieser Ordner wurde von der von TYPO3 Flow übernommen. In einer der nächsten Versionen sollen hier die Extensions gespeichert werden. Im aktuellen Fall liegen hier externe Bibliotheken, die mit Composer <sup>8</sup> installiert wurden, wie zum Beispiel Doctrine
fileadmin/	In diesem Ordner werden Dateien gespeichert, die über die Website erreichbar und ausgeliefert werden sollen. Dazu zählen CSS-, Image, HTML-Template- und TypoScriptdateien. Allgemein also Dateien, die vom Websitebetreiber hochgeladen werden.
typo3/	Der TYPO3 CMS Kern
contrib/	Bibliotheken von Drittanbietern
contrib/ext/	Das Verzeichnis für globale Extensions
gfx/	Jegliche Grafiken, die im Core verwendet werden
gfx/install/	Hier befand sich in früheren Versionen das Installtool. Aktuell existiert das Verzeichnis nur noch aus Kompatibilitätsgründen und wird in einer der nächsten Versionen entfernt. Das Installtool wurde als Systemextension realisiert und ist im entsprechenden Ordner unter <i>sysext/install/</i> zu finden
gfx/js/	Hier befinden sich die JavaScript Bibliotheken, die von Core genutzt werden.
gfx/mod/	Enthält die Konfiguration der Hauptmodule des Backends (File, Help, System, Tools, User, Web).
sysext/	Enthält die Systemextensions.
typo3conf/	Lokale Extensions und die lokale Konfiguration
typo3temp/	Temporäre Dateien
uploads/	Dateien die vom Websitebesucher hochgeladen werden - zum Beispiel über ein Formular.

Tabelle 2.1: Erläuterung der Verzeichnisstruktur von TYPO3 CMS

Version: Commit 00ad2e7 From 2014-04-24 21:08:28 +0200



- Titel der Website
- Einstellungen zum Erzeugen von Graphiken

Die Einstellungen zur Datenbank werden im praktischen Teil näher beleuchtet.

## TCA

Wie bereits geschrieben wurde, stellt das Backend (BE) eine Ansicht auf die Datenbank dar. Die Inhalte werden dabei mittels Formulare eingegeben und in der Datenbank gespeichert. Die Konfiguration dieser Formulare erfolgt über ein globales PHP Hypertext Processor-Array - dem Table Content Array (TCA).

Über das TCA werden die Metadaten einer Tabelle (Datentyp, Länge, Engine) mit weiteren Daten angereichert. So können mit dem TCA

- die Beziehungen einer Tabelle zu anderen Tabellen beschrieben werden
- in welchem Layout soll ein Feld im Formular dargestellt werden
- und wie soll das Feld validiert werden.

Enthält eine Tabelle keinen Eintrag im TCA ist sie im Backend nicht

[HINWEIS: Vielleicht kann das TCA auch einfach im Glossar beschrieben werden. Einen Eintrag gibt es schon unter tcag]

## XCLASS

TYPO3 CMS besitzt einen Mechanismus, der es erlaubt Klassen zu erweitern oder Methoden mit eigenem Code zu überschreiben. Dies funktioniert für den Systemkern wie auch für andere Extensions. Dieses Feature nennt sich XCLASS und wird vom Prototypen eingesetzt um die Datenbankklasse von TYPO3 CMS zu überschreiben. Darauf wird im Kapitel [KAPITEL Analyse Ist-Zustand einfügen] näher eingegangen. Hier soll lediglich der Hintergrund zu XCLASS beschrieben werden.

Damit eine Klasse per XCLASS erweiterbar ist, darf sie nicht per `new()` Operator erzeugt werden, sondern mit der von TYPO3 CMS angebotenen Methode `\TYPO3\CMS\Core\Utility\GeneralUtility::makeInstance()`. Diese Methode sucht im globalen PHP Hypertext Processor-Array `$GLOBALS['TYPO3_CONF_VARS']` nach angemeldeten Klassen, instanziiert diese und liefert sie anstelle der Originalklasse zurück. Dieses Array dient der Verwaltung der zu

überschreibenden Klassen und erfolgt in der Datei *ext\_localconf.php* innerhalb des Extensionsverzeichnis<sup>9</sup>.

Der Mechanismus hat jedoch ein paar Einschränkungen:

- der Code der Originalklasse kann sich ändern. Es ist somit nicht sichergestellt, dass der überschreibende Code weiterhin das macht, wofür gedacht war
- XCLASSes funktionieren nicht mit statischen Klassen, statischen Methoden und finalen Klassen
- eine Originalklasse kann nur einmal per XCLASS überschrieben werden
- einige Klassen werden sehr früh bei der Initialisierung des Systems instanziiert. Das kann dazu führen, dass Klassen die als Singleton ausgeführt sind, nicht überschrieben werden können oder es kann zu unvorhergesehenen Nebeneffekten kommen.

#### 2.1.4 Extensions

Extensions sind funktionale Erweiterungen. Sie interagieren mit dem Systemkern über die Extension API und stellen die Möglichkeit dar TYPO3 CMS zu erweitern und anzupassen.

[2. Bild von <https://typo3.org/extensions/what-are-extensions/> einfügen oder nachbauen]

Extensions werden - je nach Kontext - in unterschiedliche Kategorien eingeteilt, die hier kurz vorgestellt werden.

#### Einteilung

Systemextension werden mit dem System mitgeliefert und befinden sich ausschließlich im Ordner *typo3/sysexst/*. Sie werden nochmals unterteilt in jene, die für den Betrieb von TYPO3 CMS unabdingbar sind und solche die nicht zwangsläufig installiert sein müssen, jedoch wichtige Funktionen beisteuern. Die Extension DBAL ist in die letzte Kategorie

Neben Systemextensions gibt es noch globale<sup>10</sup> und lokale Extensions. Lokale Extensions werden im Ordner *typo3conf/ext/* und globale Extensions im Ordner *typo3/ext* installiert.

<sup>9</sup> Zur Erläuterung des Aufbaues einer Extension siehe Kapitel 2.2.

<sup>10</sup> Da globale Extensions nur in bestimmten Szenarien einen Sinn ergeben und in der Realität so gut wie nicht vorkommen, wird von der TYPO3 Community der Begriff "Extension" synonym zum Begriff "lokale Extension" verwendet. Die Arbeit folgt dieser Regelung.

Eine weitere Kategorisierung erfolgt nach dem Aufgabengebiet einer Extension. Die Festlegung auf eine der folgenden Kategorien hat keine direkte Auswirkung auf die Funktion der Extension. Sie wird von TYPO3 CMS hauptsächlich als Sortiermerkmal im Extension Manager genutzt.

[3. Bild von <https://typo3.org/extensions/what-are-extensions/> einfügen oder eigenen Screenshot machen]

- Frontend
- Frontend Plugins
- Backend
- Backend Modul
- Service
- Example
- Templates
- Documentation
- Verschiedenens

### *Extension Manager*

Der Extension Manager (EM) Extension Manager (EM) ist ein BE Modul, über das die Extensions verwaltet werden können. Es erlaubt die Aktivierung, Deaktivierung, Herunterladen und das Löschen von Extensions. Darüberhinaus bietet der EM Möglichkeiten zur detaillierten Anzeige von Informationen über die Extensions wie das Changelog<sup>11</sup>, Angaben zu den Autoren und Ansicht der Dateien der Extension.

### *Verzeichnisstruktur*

Unabhängig von der Einteilung der Extensions in die verschiedenen Kategorien unterscheiden sie sich nicht in der Verzeichnis- und Dateistruktur. Mit der Integration von Extbase in TYPO3 CMS hat sich eine neue Verzeichnisstruktur etabliert. Sie folgt dem Paradigma "Convention statt Konfiguration", was bedeutet, dass durch Einhaltung der Struktur keine weitere Konfiguration notwendig ist.

<sup>11</sup> Das Protokoll der Codeänderungen, die ein Programm von Version zu Version erlebt

```

.
├── Classes/
│   ├── Install/
│   ├── Loggers/
│   └── Persistence/
├── Configuration/
│   ├── ExtensionBuilder/
│   ├── TCA/
│   └── TypoScript/
├── Documentation/
├── Resources/
│   ├── Private/
│   └── Public/
├── Tests/
│   ├── Build/
│   └── Unit/
├── vendor/
│   ├── bin/
│   ├── composer/
│   ├── doctrine/
│   └── symfony/
├── composer.json
├── ext_emconf.php
├── ext_icon.gif
├── ext_localconf.php
├── ext_tables.php
├── ext_tables.sql
└── ext_tables_static+adt.sql

```

Abbildung 2.3: Verzeichnisstruktur einer Extension

Verzeichnis / Datei	Erklärung
Classes/	Hier erwartet TYPO3 CMS alle Klassendateien. Diese können in weiteren Unterverzeichnissen nach ihren Zweck unterteilt werden (Controller, Service, Loggers, Persistence)
Configuration/	Enthält Konfigurationsdateien wie TypoScript oder TCA.
Documentation/	Enthält die Dokumentation im ReST <sup>12</sup> Format aus der HTML und ein PDF generiert werden kann.

12 reStructuredText <http://docutils.sourceforge.net/rst.html>

Resources/Private/	Enthält die Fluid-Template <sup>13</sup> , Sprach- und Sassdateien. Kurz alle Dateien, die für die Struktur und das Aussehen einer Website notwendig sind, jedoch lediglich Templates darstellen und noch verarbeitet werden müssen
Resources/Public/	Hier liegen Grafiken, CSS- und Javascriptdateien
Tests/	Hier werden die PHPUnit-, Akzeptanz- und/oder Verhaltenstests abgelegt.
vendor/	Dieses Verzeichnis wird von Composer angelegt und enthält externe Abhängigkeiten wie in dem Fall Doctrine DBAL. Im Moment ist dies redundant, da der selbe Inhalt auch im Webroot der Site unter <i>Packages/Libraries/</i> verfügbar sein muss. Das liegt an der noch nicht vollständig umgesetzten Kompatibilität von TYPO3 CMS zu Composer, die jedoch in einer späteren Version noch nachgereicht wird.
composer.json	Seit Version 6.2 von TYPO3 CMS ist eine <i>composer.json</i> erforderlich. In ihr werden Metadaten der Extensions wie Name, Typ (System- oder lokale Extension), Lizenz, Version und Abhängigkeiten definiert. Sie wird in Zukunft wahrscheinlich die Datei <i>ext_emconf.php</i> ablösen.
ext_emconf.php	Diese Datei ist unabdinglich für die Funktionsweise der Extensions. Sie definiert ebenso wie die <i>composer.json</i> Metadaten jedoch in einem PHP Hypertext Processor-Array und nicht im JSON <sup>14</sup> -Format. Diese Datei existiert seit den Anfängen von TYPO3 CMS und wird wahrscheinlich bald von der <i>composer.json</i> abgelöst werden.
ext_icon.gif	Ein Icon für die Extension, welches im BE angezeigt wird
ext_localconf.php	In dieser Datei wird die Extension für einen Hook oder eine XCLASS registriert.
ext_tables.php	Die Datei hat drei Aufgaben:

<sup>13</sup> TYPO3s Templating Sprache

<sup>14</sup> <http://json.org/>

ext_tables.sql	Diese Datei enthält Anweisungen um eine Datenbanktabelle zu erstellen. Sie sind im MySQL Format zu formulieren, auch wenn als Datenbank etwas anderes genutzt wird. TYPO3 CMS parst die Datei mit einem eigenen (sehr rudimentären) SQL-Parser und generiert eine eigene SQL-Abfrage. Dieses Vorgehen hat den Hintergrund, dass dadurch auf Fehler in diesen Dateien reagiert werden kann und zum anderen das die mitgelieferte Datenbankabstraktionsschicht die Anweisungen in das SQL der entsprechenden Hersteller übersetzen kann. Auf dem Inhalt dieser Datei wird im praktischen Teil in Kapitel ?? [KAPITEL REFERENZ einfügen] noch genauer eingegangen.
ext_tables_static+adt.sql	Auch diese Datei enthält SQL Code wie <i>ext_tables.sql</i> . Der Unterschied besteht darin, dass sie <i>INSERT</i> Statements enthalten kann um statische Daten in eine Tabelle bei der Installation einer Extensions einzufügen. Als Beispiel sei hier der Extension Manager genannt, in dessen Tabelle wird über diese Datei die URL zum TYPO3 Extension Repository (TER) eingefügt. Auf dem Inhalt dieser Datei wird im praktischen Teil in Kapitel ?? [KAPITEL REFERENZ einfügen] noch genauer eingegangen.

Tabelle 2.2: Erläuterung der Verzeichnisstruktur einer Extension

Abgrenzung Backend / Frontent Skizze mit Backend / Frontend / Datenbank bei erwähnung der Caches. Diese jedoch sind zu vernachlässigen, da Fokus auf der Datenbank lag. Zugriff auf die Datenbank Klassische Datenbank Anwendung -> Hinten Daten rein, vorne Daten raus

## 2.2 *Doctrine 2 DBAL*

### 2.2.1 *Beschreibung*

### 2.2.2 *Abgrenzung zum ORM*

### 2.2.3 *Verbreitung*

## 2.3 *PDO*

## 2.4 *Versionskontroll- und Issuetrackingsystem*

## 2.5 *Unit Testing*

## 2.6 *Sicherheit*

## PROTOTYPISCHER NACHWEIS DER HERSTELLBARKEIT

---

### 3.1 *Refactoring der alten Datenbank API*

#### 3.1.1 *Tests für die alte Datenbank API*

### 3.2 *Testgetriebene Implementierung der neuen Datenbank API*

- Adapter - alte API Klasse erbt von neuer API Klasse - Verbindung mit der Datenbank via Doctrine - Methodenübername wenn sinnvoll
- Neue Methoden

#### 3.2.1 *Einführung von Query Objekten*

*SELECT*

*INSERT*

*UPDATE*

*DELETE*

*TRUNCATE*

### 3.3 *Anwendung der neuen Datenbank API*

### 3.4 *Überprüfen der Funktionalität*



## AUSBLICK

---

- Einladung zum ACME14N
- Einbau in den Core
- Nutzung von Doctrine Migrations
- Nutzung von Doctrine ORM
- Umbau der Datenbankobjekte zu einem Datenbankconnection Pool Design Pattern Static Fabric oder Fabric

## ZUSAMMENFASSUNG

---

## QUELLENVERZEICHNIS

---

- [Dam10] Karsten Dambekalns. *FrOSCamp 2010 in Zürich*. Englisch. Sep. 2010. url: <http://karsten.dambekalns.de/blog/froscamp-2010-in-zurich.html> (besucht am 08. 04. 2014).
- [DEM14] Karsten Dambekalns, Benjamin Eberlei und Christian Müller. *The reasons why TYPO3 Flow switched to Doctrine ORM*. microblog. Apr. 2014. url: <https://twitter.com/konafets/status/453102477081341952> (besucht am 08. 04. 2014).
- [DRB08] Dmitry Dulepov, Ingo Renner und Dhiraj Bellani. *TYPO3 extension development developer's guide to creating feature-rich extensions using the TYPO3 API*. Englisch. Birmingham, U.K.: Packt Pub., 2008. isbn: 9781847192134 1847192130 1847192122 9781847192127.
- [Ebe12] Benjamin Eberlei. *Extension that replaces TYPO3 Extbase ORM with Doctrine2*. Englisch. Apr. 2012. url: <https://github.com/simplethings/typo3-extbase-doctrine2-extension> (besucht am 09. 04. 2014).
- [Lab+06] Kai Laborenz u. a. *TYPO3 4.0: das Handbuch für Entwickler ; [eigene Extensions programmieren ; TypoScript professionell einsetzen ; barrierefreie Websites, Internationalisierung und Performancesteigerung ; inkl. AJAX und TemplaVoila]*. German. Bonn: Galileo Press, 2006. isbn: 9783898428125 3898428125.
- [Mar12] Thomas Maroschik. *WIP: Doctrine DBAL integration*. Englisch. Dez. 2012. url: <https://git.typo3.org/Packages/TYPO3.CMS.git/tree/b0a64733d2de2396c1f91dd559b37f0a4b652766> (besucht am 09. 04. 2014).
- [TYP08] TYPO3 Association. *Berlin Manifesto*. Englisch. Okt. 2008. url: <http://typo3.org/roadmap/berlin-manifesto/> (besucht am 06. 04. 2014).
- [TYP13] TYPO3 Core Team Mailingliste. *TYPO3 - Team Core - RFC: Integrate Doctrine2 into TYPO3 CMS*. Englisch. Jan. 2013. url: <http://typo3.3.n7.nabble.com/RFC-Integrate-Doctrine2-into-TYPO3-CMS-td237490.html> (besucht am 09. 04. 2014).

## ABKÜRZUNGSVERZEICHNIS

---

**API** Application Programming Interface

**BE** Backend

**CMF** Content Management Framework

**CMS** Content-Management-System

**DBAL** Database Abstraction Layer

**ECMS** Enterprise Content Management-System

**EM** Extension Manager

**FE** Frontend

**GPL2** GNU General Public License v.2

**JCR** Content Repository for Java Technology API

**MVC** Model-View-Controller

**ORM** Object-relational mapping

**PHP Hypertext Processor** PHP

**T3Assoc** TYPO3 Association

**TCA** Table Content Array

**TER** TYPO3 Extension Repository

**WCMS** Web Content Management-System

## TABELLENVERZEICHNIS

---

2.1	Erläuterung der Verzeichnisstruktur von TYPO3 CMS .	10
2.2	Erläuterung der Verzeichnisstruktur einer Extension . .	16

## ABBILDUNGSVERZEICHNIS

---

2.1	Zeitachse der TYPO3 CMS Entwicklung . . . . .	5
2.2	Verzeichnisstruktur von TYPO3 CMS . . . . .	9
2.3	Verzeichnisstruktur einer Extension . . . . .	14

## LIST OF LISTINGS

---

## EIDESSTATTLICHE ERKLÄRUNG

---

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Flensburg, den 25. April 2014

---

Stefan Kowalke