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:	24. April 2014
Erstprüfer:	Prof. Dr. Hans-Werner Lang
Zweitprüfer:	Dipl. VK Tobias Hiep

Dieses Dokument wurde am 24. April 2014 mit LaTeX 2ε gesetzt.

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

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 geparst und in die anderen SQL-Dialekte konvertiert werden kann, müssen sich die Entwickler - sei es TYPO3 Kern- oder externer Extensionprogrammier - 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 verwendert die DBAL Extension die extene Bibliothek *AdoDB* welche nicht mehr aktiv weiterentwickelt wird und weder Verbessungen 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 eine 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 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

1

EINLEITUNG

Als sich auf den Developer Days 2006 das Entwicklerteam für einen Nachfolger der eben erst erschienen TYPO3 Version 4.0 formierte (vgl. [08]), war wohl keinem der dort Anwesenden klar wohin die Reise gehen würde – ging man anfänglich noch von einem Refactoring[1] 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 Chefentwickler 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

1 Strukturverbesserung des Quellcodes bei Beibehaltung der Funktionalität

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. [08]) bekanntgegeben, welches mit kappen Worten feststellt[2]:

> "
>
> - 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 heranführt.

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-

2 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. [Quellen angabe]

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 Associationen 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.

2

GRUNDLAGEN

2.1 *TYPO3*

2.1.1 *Geschichte*

TYPO3 CMS ist ein Web Content Management-Sytem (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[1] gab es am 7. April 2014 208561 Installationen von TYPO3 CMS.

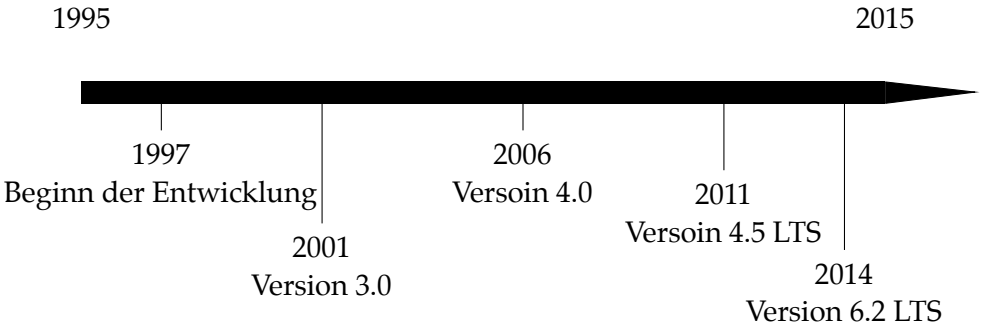

Abbildung 2.1: Zeitachse der TYPO3 Entwicklung

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

- aus TYPO3 v4[2] wurde TYPO3 CMS
- aus FLOW3 wurde TYPO3 Flow
- und aus TYPO3 5.0 / TYPO3 Phoenix wurde TYPO3 Neos

1 http://t3census.info/

2 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.[3]

2.1.2 *Definition*

TYPO3 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-Sytem (ECMS)[4], was als Hinweis auf den Einsatz des Systems für mittel- bis große Webprojekte dient.

Als letztes sei noch erwähnt, dass TYPO3 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 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/
-->
```

3 http://association.typo3.org/
4 http://www.typo3.org

2.1.3 *Architektur und Aufbau von TYPO3*

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 - basierend auf dem Konzept der Objektorientierung - geschrieben und ist damit auf jeder Platform lauffähig, die über einem PHP Interpreter verfügt. Die Version 6.2 von TYPO3 CMS benötigt mindestens PHP 5.3.7.

PHP bildet zusammen mit einem Apache Webserver und einer MySQL Datenbank den sogenannten Webstack, der abhängig von dem eingesetzten Betriebssystem MAMP (OSX / **M**ac), LAMP (**L**inux) oder WAMP (**W**indows) 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 in zwei Bereiche:

- das Backend
  stellt die Administrationsoberfläche dar. Hier erstellen und verändern Redaktuere 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])

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[5] 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 frühren TYPO3 Versionen gibt es kein "Dummy"-Package[6] mehr. Ab Version 6.2 enhält der Download lediglich den TYPO3 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 und das Verzichnis *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 *typo3_src* auf das Installationverzeichnis der neuen Version "umgebogen" werden muss.

5 CRUD - **C**reate, **R**etrieve, **U**pdate und **D**elete

6 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/
```

Im folgenden werden die einzelnen Verzeichnisse näher erklärt:

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[7] 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 Kern
contrib/	Bibliotheken von Drittanbietern
ext/	Das Verzeichnis für globale Extensions
gfx/	Jegliche Grafiken, die im Core verwendet werden
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 Sytemextension realisiert und ist im entsprechenden Ordner unter *sysext/install/* zu finden
js/	Hier befinden sich die JavaScript Bibliotheken, die von Core genutzt werden.
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.

7 Ein Kommandozeilen Programm, um Abhängigkeiten in PHP Projekten aufzulösen https://getcomposer.org/

Im Verzeichnis *www.example.com* muss noch ein Symlink *index.php* angelegt 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: `print(x**2) print(x**2)`

```
1 $GLOBALS['TYPO3_CONF_VA
2  'className' => 'Docume
3 );
```

- Debug Mode
- Sicherheitslevel für den Login (Fronend und Backend)
- das Passwort für das Installtool (mit MD5 und Salt gehasht)
- die Zugangsdaten zur Datenbank (Benutzername, Password, Datenbankname, Socket, …)
- Einstellungen zum Caching
- Titel der Website
- Einstellungen zum Erzeugen von Graphiken

Die Einstellungen zur Datenbank werden im praktischen Teil näher beleuchtet.

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 Datenbankanwendung -> 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*

3

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*

4

AUSBLICK

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

ZUSAMMENFASSUNG

[08] *Berlin Manifesto*. Englisch. Okt. 2008. url: http://typo3.org/roadmap/berlin-manifesto/ (besucht am 06. 04. 2014).

[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*. English. 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).

[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

CMF Content Management Framework

CMS Content-Management-System

DBAL Database Abstraction Layer

ECMS Enterprise Content Management-System

GPL2 GNU General Public License v.2

JCR Content Repository for Java Technology API

MVC Model-View-Controller

ORM Object-relational mapping

T3Assoc TYPO3 Association

TCA Table Content Array

WCMS Web Content Management-System

TABELLENVERZEICHNIS

ABBILDUNGSVERZEICHNIS

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 24. April 2014

Stefan Kowalke

Version: Commit: 6f7aa58 from 2014-04-22 22:52:59 +0200