

Semesterarbeit

Thema: TYPO3 heute und in Zukunft

Copyright:

Jürg Langhard
Dienstag, 1. Juni 2010

Student:

Jürg Langhard
Im oberen Gern 36
8409 Winterthur

langhard@hsz-t.ch
079 690 75 96

Dozent:

Matthias Bachmann

mbachman@hsz-t.ch

Inhaltsverzeichnis

Semesterarbeit.....	1
Editorial.....	8
Wer ist Jürg Langhard?.....	8
2003.....	8
2005.....	8
2006.....	8
2008.....	8
2010.....	8
Einleitung.....	9
Semesterarbeit	9
Ausgangslage.....	9
Aufgabenstellung.....	9
Abgrenzungen.....	9
Detailanalyse der Aufgabenstellung	9
TYPO3-Extensions „tt_news“.....	9
Reise durch die Zeit von TYPO3.....	10
Die Entstehung von TYPO3	10
1997.....	10
Am Anfang war der Test	10
1998.....	10
Prototyping	10
Kommerzielle Entwicklung	10
Marktsondierung	10
Demonstration mit David Siegel in Frankreich	10
Zusammenschluss mit Saatchi & Saatchi	10
1999.....	10
Nochmal von vorne	10
Trennung von superfish.com	11
Ein Jahr der Stille	11
2000.....	11
TYPO3 wird Open Source.....	11
Wachsende Gemeinschaft	11
2001.....	11
Interne Optimierungen.....	11
Artikel in "iX", des Lobes voll	11
2002.....	11

Snowboarding in Österreich	11
Das erste „echte“ Release	12
PHP-magazin Bericht über TYPO3	12
3.5.0 und der Extension Manager	12
TYPO3 heute.....	12
FLOW3 und Extbase.....	12
Design und Architektur.....	12
PHP-Programmierung heute.....	13
Objektorientierte Programmierung.....	13
Geschichtliches.....	13
Objekte und Klassen.....	13
Buchempfehlung.....	13
Design Patterns.....	14
Domain Driven Design (DDD).....	14
Das Domänenmodell	14
Domänenspezifische Sprache.....	14
Wichtigster Vorteil.....	14
Schichtenarchitektur.....	14
Model View Controll (MVC).....	15
Problemlösungszyklus.....	16
Ausgangslage und Verbesserungsvorschläge.....	16
Pflege von Inhalten (Ausgangslage).....	16
Pflege von Inhalt (Verbesserungsvorschlag).....	17
Frontend.....	17
Backend (Admin-Tool).....	17
Backend (List-Ansicht).....	17
Mögliche andere Varianten.....	17
Vervielfachte Datensätze.....	17
Mockups / Wireframes.....	18
Frontend-Views.....	18
Post / Latest.....	18
Post / Archiv.....	19
Post / RSS.....	20
Post / Single.....	21
Frontend-Editing.....	22
Post / Edit (FE-Editing).....	22
Comment/Edit (FE-Editing).....	23
Category/Edit (FE-Editing).....	23
Positionsbeschreibung.....	24

Vom Wireframe zur Extension.....	27
UML	27
Datenbankentwurf.....	28
Domänenanalyse	28
Arten des Modells.....	28
Entities (Entitäten).....	28
Value (Object).....	28
Service.....	28
Aggregates-Root.....	28
Anpassung des Datenbankentwurfes für Extbase	29
Entwurf der Tabellen für die neue Extension.....	29
Vorgehensweise der Umsetzung.....	30
Test-driven Development	30
FLOW3, Extbase und Fluid.....	31
Grundlagen	31
Was ist Extbase?.....	31
Für was wird Fluid verwendet?.....	31
Installation	31
Nötige Extension installieren.....	31
Installation erfolgreich.....	32
TYPO3-Seite vorbereiten	32
TypoScript Page-Objekt erstellen.....	32
Neue Seite innerhalb von TYPO3 erstellen.....	32
Neue Extension erstellen	33
Verzeichnisse und Basis-Dateien.....	33
File: ext_emconf.php.....	33
File: ext_localconf.php.....	33
Namenskonventionen.....	33
Weshalb 'kiddog_news'?	34
Registrieren der Extension innerhalb von TYPO3.....	34
FE-Plugin.....	34
Backend-Modul.....	34
Extension installieren.....	35
Konfiguration der Datenbanktabellen.....	36
Domain-Models.....	36
Post.....	36
Comment.....	36
Category.....	37
File.....	37

Link.....	37
Tag.....	37
Datenbankstruktur.....	37
File: ext_tables.sql.....	37
TCA-Anpassen.....	40
Auswahl (TYPE: „select“)	40
tca.php.....	40
Datenbank.....	41
TYPO3-Core-API.....	41
File-Upload (TYPE: „group“, „file“)	41
tca.php.....	41
Text-Eingabefeld mit RTE	42
tca.php.....	42
Controller.....	43
Controller in Extbase	43
Vererbungsdiagramm der Controller-Klassen.....	43
Beispiel-Controller.....	44
Übergabe von Paramter(n) an einen Controller.....	44
Beispiel: showSingleAction().....	45
Verwendete Controller	45
PostController.....	45
CommentController.....	45
FileController.....	45
CategoryController.....	45
LinkController.....	45
View.....	46
Beispiel View (showlatest.html)	46
Validierung von Eingaben.....	47
Beispiel: Attribut „Title“ des Domain-Mode Post.....	47
Validierung = false.....	47
Validationsmöglichkeiten in Extbase	47
Alphanumerik	48
EmailAddress	48
NotEmpty	48
NumberRange	48
RegularExpression	48
StringLength	48
Domain/Repository.....	49
Funktionalität	49

Auslesen von Objekten.....	49
Manipulieren von Objekten.....	50
Beispiele.....	50
Erweitern der Repository-Klasse.....	50
Verwendetet Repositories.....	50
Schlusswort.....	51
Weiterführende Informationen.....	51
Ziel.....	51
Anhang I: Herkömmliche Extension.....	52
Erstellung mittels Kickstarter.....	52
General Info.....	52
Auszug aus ext_emconf.php.....	52
Setup languages.....	53
New Database Tables und Extend existing Tables.....	53
Frontend- Backend Plugins.....	54
Frontend-Plugin.....	54
Backend-Modul.....	54
Static TypoScript code / Tsconfig.....	54
Fertigstellen und laden der Erweiterung.....	54
Erweiterung wurde erfolgreich installiert.....	54
Anhang II: Ajax im Backend.....	55
Filesystem.....	55
Javascript Framework.....	55
Einbinden von ExtJSmit Fluid.....	55
Ajax-Methode registrieren.....	56
Aufruf-Beispiel.....	56
Anhang III: TypoScript-Parameter.....	57
Konfigurationen mittels TypoScript.....	57
Anhang IV: Mini-Cheat-Sheet Fluid.....	58
Wichtige Funktionen.....	58
Texte kürzen (f:format.crop).....	58
Bild rendern (f:image).....	58
Verlinkungen.....	58
Action.....	58
Files.....	58
E-Mail.....	58
Externe URL.....	58
Formulare.....	58
Select (Array).....	58

Fehlermeldungen.....	59
RuntimeException.....	59
Duplicate variable declarations.....	59
Tx_Fluid_Core_Parser_Exception.....	59
Cannot cast object of type "DateTime" to string.....	59
Required argument "src" was not supplied.....	59
Quellenverzeichnis.....	60
Bücher.....	60
Webdatenbank-Applikationen mit PHP und MySQL.....	60
TYPO3 4.0: Das Handbuch für Entwickler.....	60
Certified TYPO3 Integrator	60
TYPO3-Extensions.....	60
PHP objektorientiert.....	60
PHP Design Patterns	60
UML 2 glasklar	60
Entwurfsmuster von Kopf bis Fuss.....	60
Zeitschriften.....	61
t3n .open .web .business.....	61
Nr. 18: Kategorien „Content Management“	61
Vortrags- und MicroDokumentationen.....	61
Get into the FLOW with Extbase.....	61
MVC for TYPO3 4.3 with extbase.....	61
Workshop Extension-Entwicklung mit Extbase und Fluid.....	61
Fluid - The Zen of Templating.....	61
Online.....	61
TYPO3-Core-API.....	61

Editorial

Wer ist Jürg Langhard?

Gerne möchte ich an dieser Stelle ein paar Worte über meine Person und meine Arbeit mit TYPO3 verlieren, damit Sie die Möglichkeit haben mich ein wenig besser kennen zu lernen.

2003

Das erste Mal als ich mit TYPO3 in Kontakt kam war irgendwann im Jahr 2003. Zu diesem Zeitpunkt war ich Handball-Profi, spielte erfolgreich in der höchsten Liga der Schweiz bei den Vereinen Pfadi-Winterthur, St. Otmar St. Gallen und Yellow Winterthur, und widmete mich in meiner Freizeit der Erstellung von statischen Internetseiten mit HTML und CSS.

Schon sehr bald kam dann aber der Wunsch, Inhalte über ein CM-System ändern zu können. Obwohl ich zu diesem Zeitpunkt noch nicht einmal wusste, was ein CMS ist, suchte ich nach Möglichkeiten mein Wunsch umsetzen zu können. Nach ein paar unbefriedigenden Versuche von lokalen sowie von kleinen CM-Systemen landete ich am Ende bei TYPO3.

2005

Im Sommer 2005, entschied ich mich dann den Profisport an den Nagel zu hängen, und lies mich als Webpublisher beim regional TV-Sender TeleZüri anstellen. In diesem Umfeld durfte ich viele Erfahrungen sammeln, unter anderem, wie das Open Source CMS TYPO3 im professionellen und kommerziellen Umfeld zum Einsatz kommen kann.

2006

Nach einem Jahr wollte ich mehr wissen über Informatik im Allgemeinen und speziell über alles, was mit Internet zu tun hat. Nach einer längeren Suche nach einer passenden Schule entschloss ich mich an der Hochschule für Technik in Zürich, der HSZ-T, das Studium „Bachelor of Science Informatik“ berufsbegleitend zu absolvieren.

2008

Im Frühling 2008 bekam ich dann die Chance bei der Firma Digicom digitale Medien AG in Effretikon den Bereich „Webdevelopment“ zu leiten.

2010

Seit dem 22.04.2010 bin ich Inhaber und Geschäftsführer von GreenBanana GmbH.

GreenBanana GmbH

Im oberen Gern 36
8409 Winterthur



GreenBanana GmbH erbringt diverse Dienstleistungen im Internetbereich, insbesondere Konzeption, Entwicklung, Wartung und Gestaltung von Internetauftritten und internetbasierten Applikationen auf Basis von TYPO3.

Einleitung

Semesterarbeit

Ausgangslage

Das ich mich für das Thema TYPO3 sowie der zukunftssicheren Programmierung mit TYPO3 entschieden habe hat verschiedene Gründe. Zum einen wollte ich mich vertieft mit den neuen Programmiertechniken auseinandersetzen, welche das neue PHP-Framework FLOW3 (voraussichtlicher Release Sommer 2011) mit sich bringt, zum anderen möchte ich der TYPO3-Community als Dankeschön für die tolle Arbeit, die jeden Tag geleistet wird, etwas zurück geben in dem ich die Semesterarbeit nach Abschluss zur freien Verfügung auf meiner Firmen-Homepage zum Download anbieten werde.

Aufgabenstellung

Ziel dieser Arbeit ist die Programmierung eines Prototypen einer TYPO3-Erweiterung, welcher im Funktionsumfang der populären News-Extension tt_news ähnlich wird. Sie soll als Beispiel dienen um einen grossen Teil der Neuerungen, welche das neue TYPO3 mit sich bringt aufzuzeigen.

Das Hauptaugenmerk liegt jedoch ganz klar bei der Dokumentation der zukunftssicheren Programmierung, der Beschreibung der neuen Technologien sowie einer ausführlichen Anleitung zum Umgang von Extbase und Fluid.

Abgrenzungen

Diese Dokumentation hat nicht den Anspruch ein Nachschlagewerk für FLOW3, Extbase oder Fluid zu sein. Der Leser sollte mit dem Vorwissen von klassischer TYPO3-Extension-Entwicklung jedoch in der Lage sein, die neuen Programmierkonzepte und Funktionalitäten zu verstehen und anwenden zu können.

Detailanalyse der Aufgabenstellung

Da das Hauptaugenmerk auf der Dokumentation und der Erläuterung liegt, musste eine geeignete Erweiterung gesucht werden, an welcher die neuen Technologien gezeigt werden können. Ich entschloss mich, „tt_news“ zu nehmen.

TYPO3-Extensions „tt_news“

Seit der TYPO3-Version 3.5, welche im Jahr 2002 veröffentlicht wurde, ist der Extension-Manager zu einem zentralen Teil von TYPO3 geworden. Damit lässt sich das Basis-System mit individuellen Funktionalitäten erweitern, um so den Bedürfnissen der Gesellschaft gerecht werden zu können.

Mittlerweile gibt es auf der Website: www.typo3.org/extensions über 4'300 Erweiterungen, die von der Community programmiert wurden und unter der GPL-Lizenz frei genutzt und erweitert werden dürfen.

Eine der Bekanntesten ist sicherlich die News-Extension mit dem Extensions-Key „tt_news“. Sie wurde bis heute (11.04.2010) 293'973-mal aus dem Extension-Repository geladen. Mit ihr lassen sich News-Einträge verwalten und die Darstellung im Frontend steuern.

Trotz ihrer Popularität ist der verwendete Programmierstil veraltet und wird ab der TYPO3-Version 5.0 nicht mehr unterstützt.

Reise durch die Zeit von TYPO3

Die Entstehung von TYPO3

1997

Noch bevor „Content Management“ zu einem Begriff wurde, begann Kasper Skårhøj mit der Entwicklung von TYPO3.

Die heutzutage fast unübersichtliche Anzahl an Content Management Systemen entstand aus dem Bedürfnis die steigende Komplexität des Internets in den Griff zu bekommen. Die Trennung von Design und Content wurde unvermeidlich.

Am Anfang war der Test

Als Erstes wurden die Grundanforderungen für ein CMS herausgearbeitet. Der Kundenwunsch war ein Werkzeug, mit dem Texte auf einer Website selbständig und ohne HTML-Kenntnis verwaltet werden können.

1998

Prototyping

Frühling 1998: Drei Prototypen wurden programmiert, jeder etwas ausgereifter als sein Vorgänger. Dabei wurden wertvolle Erkenntnisse gewonnen, die in das finale Konzept einfließen.

Kommerzielle Entwicklung

Sommer 1998: Zur kommerziellen Entwicklung wurde TYPO3 der Webagentur Superfish.com anvertraut und strategische Ziele für die Zukunft definiert.

Marktsondierung

August 1998: Bei einem Besuch der Seybold Konferenz, bei der es unter anderem zu einem Treffen mit dem legendären Internet-Guru David Siegel kam, wurden Marktchancen ausgelotet.

Demonstration mit David Siegel in Frankreich

Oktober 1998: Nach Monaten intensiver Entwicklung wurde TYPO3 auf einem Workshop David Siegels auf der IFRA-Ausstellung in Lyon (Frankreich) als Unternehmenslösung für Content Management vorgestellt.

Zusammenschluss mit Saatchi & Saatchi

November 1998: Die Perspektiven erweiterten sich, Superfish.com ging mit der dänischen Dependance von Saatchi & Saatchi zusammen und brachte das Web-Business mit in die Ehe ein.

1999

Nochmal von vorne

Januar 1999: Basierend auf den Erfahrungen von Kunden wie Kilroy Travels, GreenSquare und einem dänischen Pionier für Stadtportale wurden die technischen Grundlagen von TYPO3 neu definiert und ein neuer, beständiger Kern von Grund auf entwickelt. Dieser ist seitdem die flexible und mächtige Grundlage für das heutige System.

Trennung von superfish.com

Sommer 1999: Nachdem Kasper Skårhøj erkannte, dass superfish.com sich anderweitig orientieren würde als in Richtung Content Management, kam es zu einem Deal mit Superfish.com. Er verliess die Firma, was ihm alle Rechte einbrachte, TYPO3 in Eigenregie weiter zu entwickeln. Ein wichtiger Punkt wurde ihm damals klar: Dass es schwierig ist, die eigenen hohen Qualitätsstandards mit der kommerziellen Software-Entwicklung zu vereinbaren. Denn, um mit der Entwicklung Schritt halten zu können, wäre man gezwungen, unausgereifte Versionen auf den Markt zu werfen, was den Quellcode sukzessive zu Stück- und Flickwerk degradieren täte. Aber genau dieses sollte mit TYPO3 nicht passieren.

Ein Jahr der Stille

August 1999: Kasper stürzte sich allein in das Projekt, um das zu beenden, was er begonnen hatte. Er veranschlagte dazu 6 Monate.

2000

Es wurden doppelt so viele. Eine Woche vor seiner Heirat mit Rie veröffentlichte er die erste Beta Version im August 2000.

TYPO3 wird Open Source

Aus der Stille der Entwicklerstube gelangte TYPO3 in die nächste Phase, direkt in die Crashtests der Open Source Welt.

Wachsende Gemeinschaft

Dezember 2000: Nach einigen Monaten in der Öffentlichkeit hatte TYPO3 eine Reihe von Qualitätstests bestanden und eine ständig wachsende Gemeinschaft hinter sich geschart. Zusätzlich befeuert wurde die Entwicklung durch neue Ideen und die Nachfragen der User nach neuen Funktionalitäten.

2001

Interne Optimierungen

Sommer 2001: Grosse Aufräum- und Optimierungsarbeiten am Code. Das Ergebnis dieser Selbstevaluierung war, dass die Codebasis von TYPO3 durch das wohldurchdachte Erweiterungskonzept mit den Extension noch robuster wurde.

Artikel in "iX", des Lobes voll

September 2001: Nachdem TYPO3 seine Überlebensfähigkeit bewiesen hatte, fand es seinen Weg in die allgemeine Wahrnehmung als ein ernstzunehmendes PHP-basiertes CMS. Dies bestätigte ein Artikel des deutschen IT-Magazins „iX“, das TYPO3 in einem Atemzug mit einem anderen bekannten Oss-CMS nannte: Zope. TYPO3 fand darin allgemeine Würdigung (bis auf ein paar Kommentare bezgl. Unzulänglichkeiten, die nun der Vergangenheit angehören).

2002

Snowboarding in Österreich

Winter 2002: Was als verrückte Idee ihren Anfang genommen hatte, endete als Seminar im österreichischen Schnee mit 25 Teilnehmern. Es war das erste Mal, dass die Entwickler persönlich einander begegneten. Das Resultat: Synergie hoch drei - eine Woche später kehrten alle voller Zuversicht nach Hause zurück.

Das erste „echte“ Release

24. Mai 2002: Bis zu diesem Tag war TYPO3 im „Beta-Test“ (obwohl es auch für die Produktion eingesetzt wurde). Mit TYPO3 Version 3.0 war ein Meilenstein erreicht.

PHP-magazin Bericht über TYPO3

29. Mai 2002: Nur wenige Tage nach der Final Release brachte das deutsche PHP-Magazin einen Bericht über PHP-basierte CM-Systeme. 40 Systeme wurden miteinander verglichen, wobei TYPO3 detailliert besprochen und mit 4 weiteren kommerziellen Systemen verglichen wurde - als einziges CMS unter der GPL stand TYPO3 in einer Reihe mit den kommerziellen Systemen im Test!

3.5.0 und der Extension Manager

Kurz nach der 3.0 Version konzipierte Kasper den Extension Manager. Er wurde im Herbst 2002 getestet, verbessert und entwickelte sich gewissermassen zu einer Art Bank für die Zukunft von TYPO3.

Zusammen mit dem Relaunch von typo3.com und typo3.org wurde Mitte November auch Version 3.5.0 zusammen mit einer neuen Corporate Identity veröffentlicht.

TYPO3 war geboren!

TYPO3 heute

FLOW3 und Extbase

Im Vergleich zur Vergangenheit werden heute zunehmend komplexere Geschäftsapplikationen innerhalb von TYPO3 realisiert. Um diesem Anspruch gerecht zu werden, wurde vom Entwicklerteam der TYPO3-Version 5.0 schon recht früh festgelegt, dass diese und zukünftige Versionen auf einem komplett neuen Gerüst aufsetzen sollen.

Dieses neue Gerüst wurde auf den Name FLOW3 getauft und ist ein Enterprise Application Framework, welches auch für beliebig andere – von TYPO3 unabhängige – Anwendungen verwendet werden kann.

Während den Transition Days im Oktober 2008 wurde das sogenannte Berliner Manifest aufgestellt, welches besagt dass TYPO3 v4 und TYPO3 v5 eine gewisse Zeit parallel weiter entwickelt werden. Dies bedeutet zum einen, dass viel verwendete Erweiterungen von v4 nach v5, aber auch Neuerungen von v5 nach v4 portiert werden.

Design und Architektur

Die Anforderungen an die heutigen Softwareprojekte sind über die Jahre kontinuierlich gestiegen. Damit die Entwickler in diesen komplexen Aufgaben nicht den Überblick verlieren, haben sich in der Vergangenheit einige hilfreiche Programmier-Konzepte herauskristallisiert.

Mit dem Domain-Driven Design und der Model-View-Controll Architektur helfen gleich zwei solche Konzepte - in Extbase und FLOW3 - auch komplexen Anforderungen gerecht zu werden.

PHP-Programmierung heute

Objektorientierte Programmierung

Geschichtliches

Das Konzept der objektorientierten Programmierung wurde bereits in den 1960er Jahren entwickelt und gehört seit Anfang der 1990er Jahre zu einem der Grundkonzepte der Softwareentwicklung.

Während viele höhere Programmiersprachen, wie C++ und Java, das Konzept bereits frühzeitig vollständig implementierten, wurde das Thema in PHP lange Zeit eher vernachlässigt. Ein umfassendes Modell zum objektorientierten Programmieren steht erst seit der PHP-Version 5, welche im Jahr 2004 erschien, zur Verfügung.

Ziele der Objektorientierung sind die klareren Strukturierungsmöglichkeiten und die einfachere Wiederverwendbarkeit von Programmquelltexten, und somit eine Reduzierung des Entwicklungsaufwandes und höherer Persistenz.

Objekte und Klassen

Die grundlegende Idee hinter der Objektorientierung ist es, gleichartige Daten in einem sogenannten Objekt zusammenzufassen. Solche Objekte sind nach aussen hin gekapselt, sodass auf die Daten in dem Objekt nur über definierte Schnittstellen zugegriffen werden kann.

Um gleichartige Objekte besser verwalten zu können, werden sogenannte Klassen als Vorlage für solche Objekte verwendet. Ein Objekt, welches aus solch einer Klasse erstellt wird, heisst dann Instanz dieser Klasse. Jedes Objekt hat bestimmte Eigenschaften, die in der Fachsprache Attribute genannt werden. Ausserdem können mit einem Objekt Aktionen durchgeführt werden, die sogenannten Methoden.

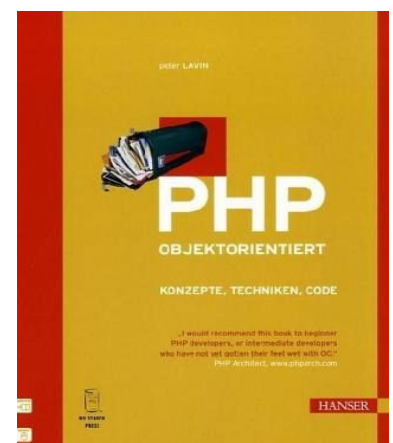
Eines der klassischen Beispiele wäre eine Klasse mit dem Namen Auto. Ein Auto zeichnet sich durch diverse Eigenschaften aus, wie zum Beispiel der Marke, Farbe, Alter, Kilometerstand, Tankfüllung und vielem mehr. Gleichzeitig können Aktionen mit dem Auto durchgeführt werden: Wir können damit fahren, es auftanken, umlackieren, und so weiter. Die Klasse „Auto“ ist jedoch nur eine abstrakte Vorlage für bestimmte Objekte. Eine Instanz der Klasse „Auto“ könnte also ein ganz bestimmtes Auto sein, also zum Beispiel ein weisser VW-Golf, Baujahr 1999, mit einem Kilometerstand von 150.000 Kilometern und einem Tankfüllstand von 15 Litern.

Buchempfehlung

Eine detaillierte Einführung in die objektorientierte Programmierung mit PHP würde leider den Rahmen dieser Semesterarbeit sprengen.

Für weitergehende Informationen rund um OOP möchte ich an dieser Stelle das deutsche Buch „PHP OBJEKTORIENTIERT – Konzepte, Techniken, Code“ von Peter Lavin empfehlen. Es ist eine kompakte Einführung in die objektorientierte Programmierung mit PHP und es werden auf einfache und kompakte Art Themen behandelt wie Klassen, Konstruktoren und Destruktoren, Vererbung und Polymorphie.

ISBN-13: 978-3-446-40762-2



Design Patterns

Die wirkliche Herausforderung bei komplexen Softwareprojekten liegt nicht in der technischen Realisierung, sondern in der genauen Umsetzung der Geschäftsprozesse und der Anforderungen der (Fach-)Nutzer.

Domain Driven Design (DDD)

Domain-driven-Design stellt konsequent die Domäne, das heisst die geschäftlichen und fachlichen Aufgabenstellungen beim Kunden, ins Zentrum des Projekts. Entscheidend dabei ist, dass das Entwicklungsteam die Domäne umfassend versteht und mit den Anwendern, statt im IT-Jargon, geschäfts- und fachorientiert kommuniziert. Basis von Domain-driven-Design ist daher eine gemeinsame, domänenspezifische Sprache und ein Domänenmodell.

Das Domänenmodell

Das Domänenmodell beschreibt mit objektorientierten Konzepten Elemente, Struktur und Prozesse der Domäne. Es bildet die relevanten Geschäfts- und Fachzusammenhänge aufgabenbezogen nach und ist der – wiederverwendbare – Kern der neuen Lösung. Das Domänenmodell wird iterativ aufgebaut, das heisst in mehreren Entwicklungszyklen zunehmend verfeinert und erweitert. So reflektiert das Modell die geschäftlichen und fachlichen Prozesse immer besser und vollständiger.

Domänenspezifische Sprache

Eine domänenspezifische Sprache ist eine formale Sprache zur Beschreibung eines geschäftlichen oder fachlichen Aufgabenfelds. Sie ist unabhängig von Programmiersprachen und wird verwendet, um die Anforderungen und die geplanten Softwarefunktionen in den Begriffen der Nutzer zu formulieren. Damit ermöglicht sie einen intensiven und effizienten Austausch zwischen Domänenexperten, Anwendern und Entwicklern.

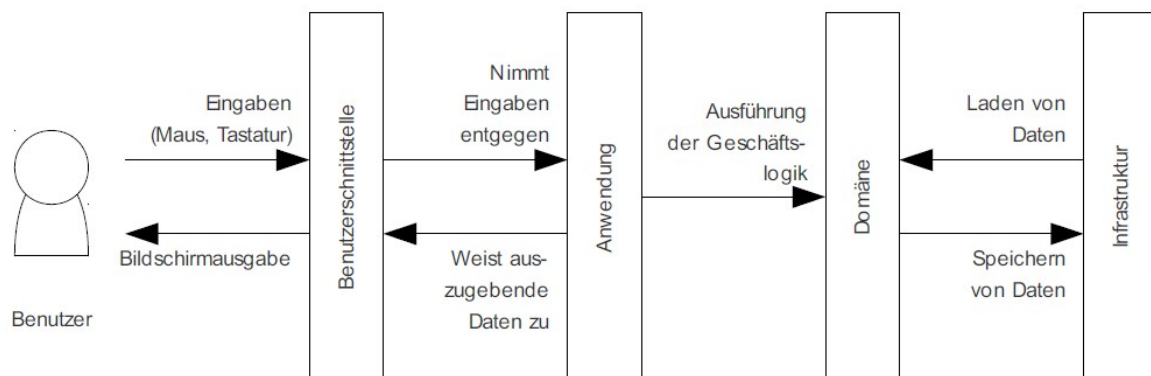
Wichtigster Vorteil

Der wichtigste Vorteil von Domain-driven-Design liegt in der anforderungsorientierten Darstellung der Geschäfts- und Fachlogik (Domänenmodell) in einer nutzerorientierten Common Language (domänenspezifische Sprache). Das Ergebnis sind schnellere Analyse-, Design- und Entwicklungsprozesse und vor allem bedarfsgerechtere Softwarelösungen. Diese Lösungen bilden die Domäne des Kunden adäquat ab, sind änderungsfreundlich und lassen sich gut weiterentwickeln.

Schichtenarchitektur

Beim Domain-Driven Design wird die Verwendung einer Schichtenarchitektur vorausgesetzt. Es werden folgende Schichten unterschieden:

1. *Benutzerschnittstelle*
Die Benutzerschnittstelle ist die Schnittstelle zwischen Benutzer und Anwendung und ist sowohl für die Darstellung am Bildschirm wie auch für die Interaktionen (Eingaben: Maus, Tastatur) verantwortlich.
2. *Anwendung*
Diese Schicht enthält keine Geschäftslogik. Sie koordiniert lediglich die Eingaben der und die Rückgabe zur Benutzerschicht.
3. *Domäne*
Das Domänenmodell ist das Herz der Anwendung. Es bildet eine Domäne der realen Welt ab und enthält die gesamte Geschäftslogik.
4. *Infrastruktur*
Die letzte Schicht ist für die technische Infrastruktur der Anwendung verantwortlich und beinhaltet zum Beispiel die Speicherung von Objekten in einer Datenbank.



Model View Controll (MVC)

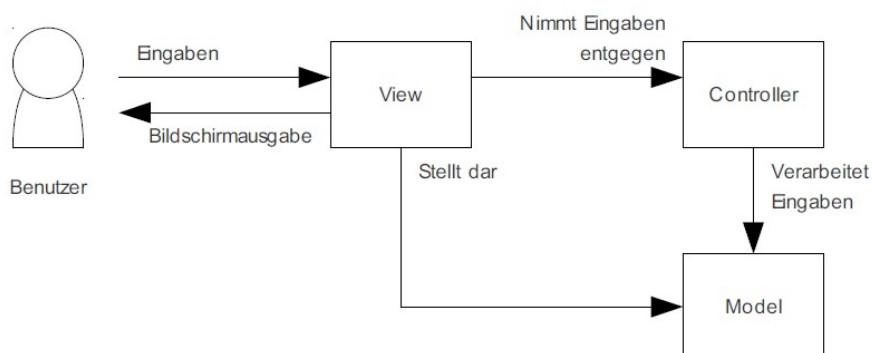
MVC mit den drei Segmenten Model, View, Controller ist ein bewährtes Design-Pattern für die Mensch-Maschine-Interaktion.

Unabhängig von Programmiersprachen, ist MVC ein De-facto-Standard für die abstrakte Beschreibung und den Grobentwurf komplexer Softwaresysteme. Die Funktionsbereiche der drei Segmente sind:

- *Model (Modell):*
Datenmodell – enthält die darzustellenden Daten und gegebenenfalls die Geschäftslogik
- *View (Präsentation):*
Präsentationssegment – stellt die Daten des Modells dar und nimmt Nutzeraktionen entgegen
- *Controller (Steuerung):*
Programmsteuerung – verwaltet die Views, empfängt von ihnen Nutzeraktionen, wertet diese aus und beantwortet sie

Aufbauend auf dieser Architektur lassen sich wiederverwendbare Komponenten mit je abgegrenzter Funktionalität entwickeln. Dabei kann und sollte die Dreiteilung nicht in jedem Fall strikt eingehalten werden. So kann es sinnvoll sein, die Validierung der Nutzereingaben im View, im Controller oder auch im Model vorzunehmen. GUI-Steuerelemente wiederum, wie Auswahllisten oder Filter, können Merkmale der Präsentation und des Controllers in sich vereinen.

Wichtigster Vorteil von MVC ist, dass das Pattern ein flexibles Programmdesign mit wiederverwendbaren Komponenten ermöglicht. Auf MVC-Basis lassen sich daher wartungs-, änderungs- und erweiterungsfreundliche Lösungen für komplexe Aufgabenstellungen realisieren.



Problemlösungszyklus

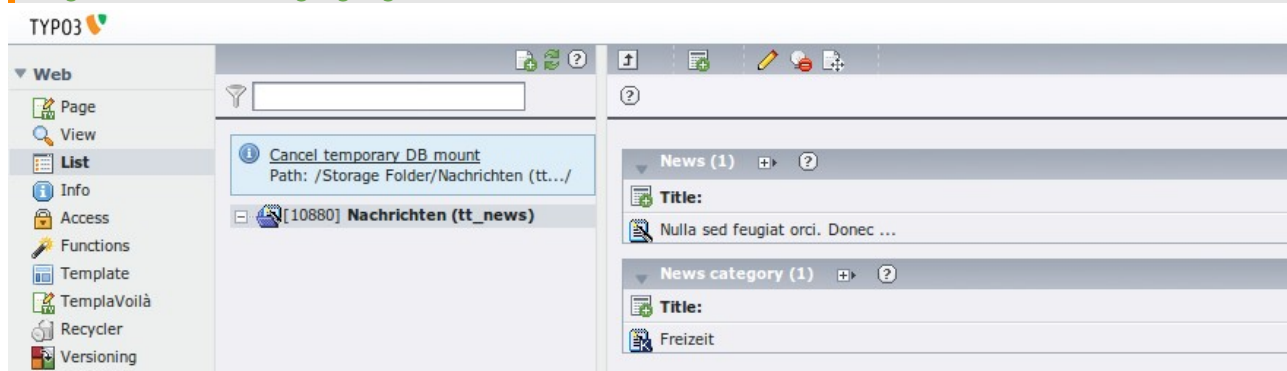
Ausgangslage und Verbesserungsvorschläge

Wie im Vorwort erläutert, stellt die Extension „tt_news“ die Grundlage für den praktischen Teil dieser Arbeit dar. Es handelt sich dabei um eine Erweiterung von TYPO3, mit der es den Redakteuren ermöglicht wird, Inhalte sowohl zeitlich wie auch kategorisch zu erfassen, verwalten und anzeigen zu lassen.

Da ich diese Erweiterung bereits mehrfach in Projekten eingesetzt habe, wusste ich wo die Schwachstellen in der Usability sind, und versuchte diese anhand genauer Mockups zu beheben.

Da die Aufgabenstellung so ist, dass die Funktionalität der Extension nicht erweitert, sondern lediglich nachgebildet werden soll lag die Herausforderung hauptsächlich darin, die Funktionalität der komplexen Erweiterung mit zukunftsicheren Methoden (MVC undDDD) sowie einer massiven Verbesserung der Usability zu programmieren.

Pflege von Inhalten (Ausgangslage)



Die Verbesserung der Usability soll damit gewährleistet werden, dass die Redakteure mehrere Möglichkeit haben News und Kategorien zu erfassen, editieren oder zu löschen und zum anderen sollen die Inhalte intuitiv im System gefunden werden.

In tt_news Version < 3.0 gab es keine Backend-Administration im eigentlichen Sinne die dem Redakteur eine übersichtliche Ansicht aller News-Datensätze zur Verfügung stellte. Die Datensätze wurden lediglich in einen dafür gekennzeichneten System-Ordner (ein sogenannter Sysfolder - im Screenshot mit der ID 10880 zu sehen) abgelegt und mussten über die TYPO3 Listen-Ansicht (List) bearbeitet werden.

Die List-Ansicht ist grob gesehen eine optisch aufbereitete Ansicht der Datenbank. Eine Art phpMyAdmin-Ansicht, welche für Redakteure verständlich gemacht wurde.

Über diese Ansicht lassen sich mit den nötigen Rechte alle Datensätze innerhalb von TYPO3 einsehen und bearbeiten, egal ob Inhaltselemente, Konfigurationen oder Seiten. Obwohl sie von erfahrene TYPO3-User sehr oft und gern verwendet wird, ist sie vor allem für Personen welche nicht täglich mit TYPO3 arbeiten meist zu abstrakt.

Ein weiterer Punkt ist der, dass eine Stärke der News-Erweiterung da liegt, dass die zentral erfassten News an verschiedenen Orten auf der Homepage ausgegeben werden können. So werden aktuelle Inhalte nach einer bestimmten Zeit nicht mehr auf der Seite „Latest“ (für die neusten Nachrichten) sondern nur noch unter „List“ (News-Liste) erscheinen. Dasselbe gilt für die automatisch Verschiebung von „List“ (News-Liste) in „Archiv“.

Diese automatischen Verschiebungen von Inhalten ist für Redakteure oftmals unverständlich, da sie nur im Frontend geschehen, die Datensätze im Backend aber immer noch im selben News-System-Folder liegen.

Pflege von Inhalt (Verbesserungsvorschlag)

Die Redakteure sollen individuell entscheiden können, ob sie die News-Einträge im Frontend, im Backend über ein übersichtliches Admin-Tool oder wie bisher in der List-Ansicht verwalten möchten.

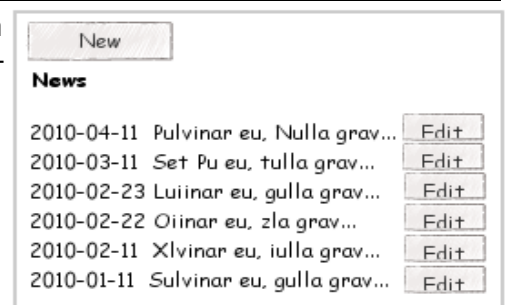
Frontend

Sobald sich ein Frontend-User am System angemeldet hat und einer Benutzergruppe angehört (welche die Berechtigung besitzt News-Einträge zu bearbeiten), soll in allen Ansichten ein Button erscheinen, welcher das Editieren des Eintrages ermöglicht.



Backend (Admin-Tool)

Im Backend soll es einen eigenen Menü-Punkt geben, über welchen alle News-Einträge, Kategorien und Kommentare hierarchisch dargestellt und editiert (erstellen, ändern, löschen) werden können.



Backend (List-Ansicht)

Weiterhin soll die Listen-Ansicht zur Verfügung stehen. Die Datensätze sollen aber neu so aufbereitet werden, dass Relationen mittels AJAX Anweisungen auf einen Blick ersichtlich sind.

So soll beim Bearbeiten eines News-Eintrages nicht nur die Felder des New-Datensatzes angezeigt werden, sondern auch gleich alle relationalen Datensätze wie Kategorien, Kommentare sowie angehängte Bilder und Files.

Mögliche andere Varianten

Vervielfachte Datensätze

Lange habe nachgedacht, ob ich anstelle von einem zentralen System-Ordner, welche alle News-Datensätze enthält dezentral arbeiten soll. So, dass jeder Seite, welche eine News im Frontend anzeigt, auch denjenigen News-Datensatz beinhaltet.

Vorteil

Für den Redakteur hätte das den Vorteil, dass die News im Backend an derselben Stelle gefunden würden, wie sie auch im Frontend angezeigt werden.

Nachteil

Durch dieses Verfahren wäre eine massive Mehrbelastung auf dem Webserver entstanden. Datensätze hätten automatisch gelöscht und verschoben, sowie mehrfach redundant in der Datenbank abgespeichert werden müssen.

Ein weiterer Nachteil wäre dabei für versierte News-Redakteure entstanden, die gerne in der Listen-Ansicht arbeiten. Dadurch, dass die News-Einträge nicht mehr zentral in einem System-Ordner abgelegt würden, wäre es fast unmöglich die Übersicht zu behalten.

Mockups / Wireframes

Nach der genauen Analyse der Ausgangslage und den erweiterten Anforderungen hat es sich bewährt, mit Mockups/Wireframes zu arbeiten.

Die nachfolgenden Mockups zeigen die möglichen Abläufe und Berechtigungen des News-Systems im Frontend und dem Frontend-Editing. Layout-Darstellungen sind nicht Gegenstand von Mockups und sind somit mit Vorsicht zu geniessen.

Frontend-Views

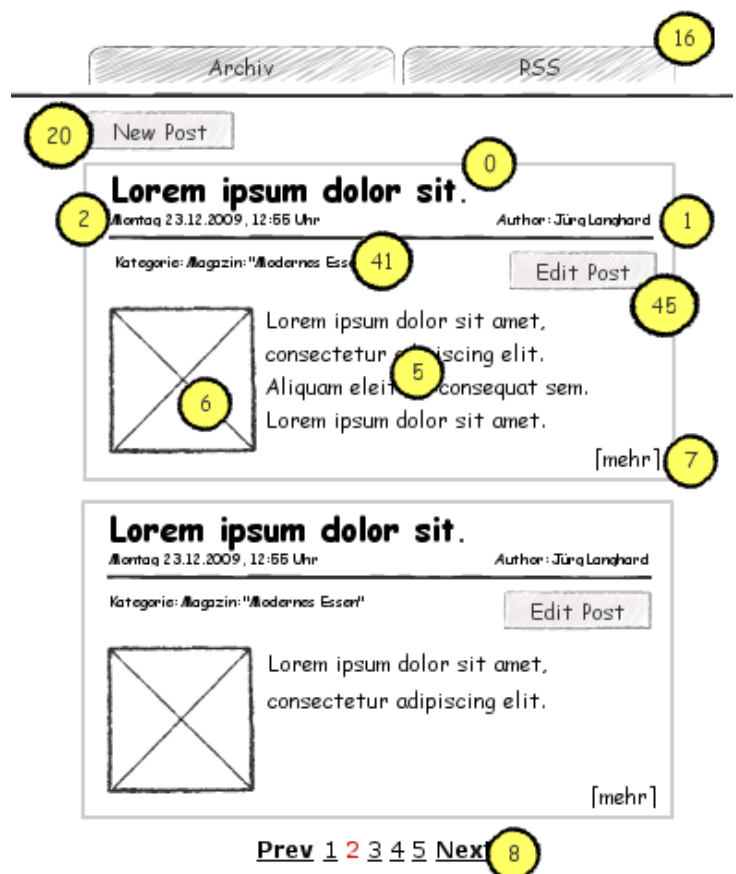
Das News-Plugin soll mehrere Frontend-Ansichten haben. Diese werden je nach übergebenem GET-Parameter angezeigt. Die Eigenschaften/Funktionalitäten werden nachfolgend erläutert und dienen später als Ausgangslage für die Definition des Klassen- und Domain-Modell.

Post / Latest

Generell

Die Latest-Ansicht bietet die Möglichkeit eine frei wählbare Anzahl von News-Einträge in einer Liste hierarchisch anzeigen zu lassen.

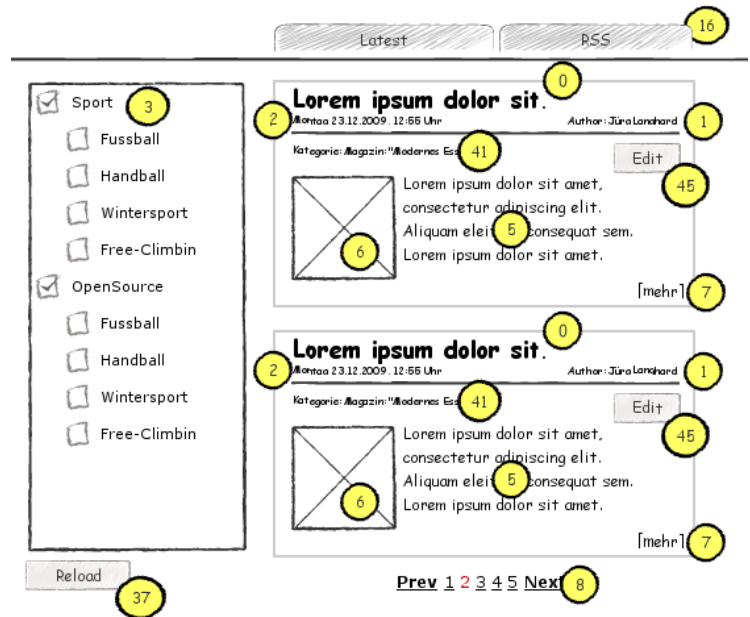
Die Positionsbeschreibungen aller Mockups befinden sich am Ende dieses Kapitels.



Post / Archiv

Generell

Die Archiv-Ansicht bietet die Möglichkeit, alle Einträge der gewählten Kategorien hierarchisch darstellen zu lassen.

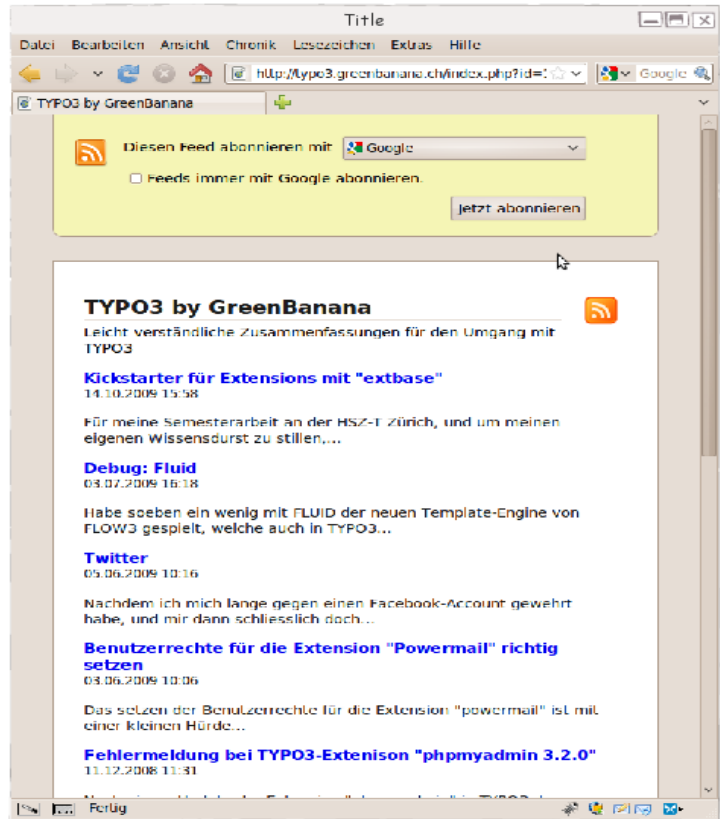


Post / RSS

Generell

Das News-System ist mit einem RSS-Feed ausgestattet. Dadurch lassen sich die News abonnieren und die Botschaft kann breiter gestreut werden.

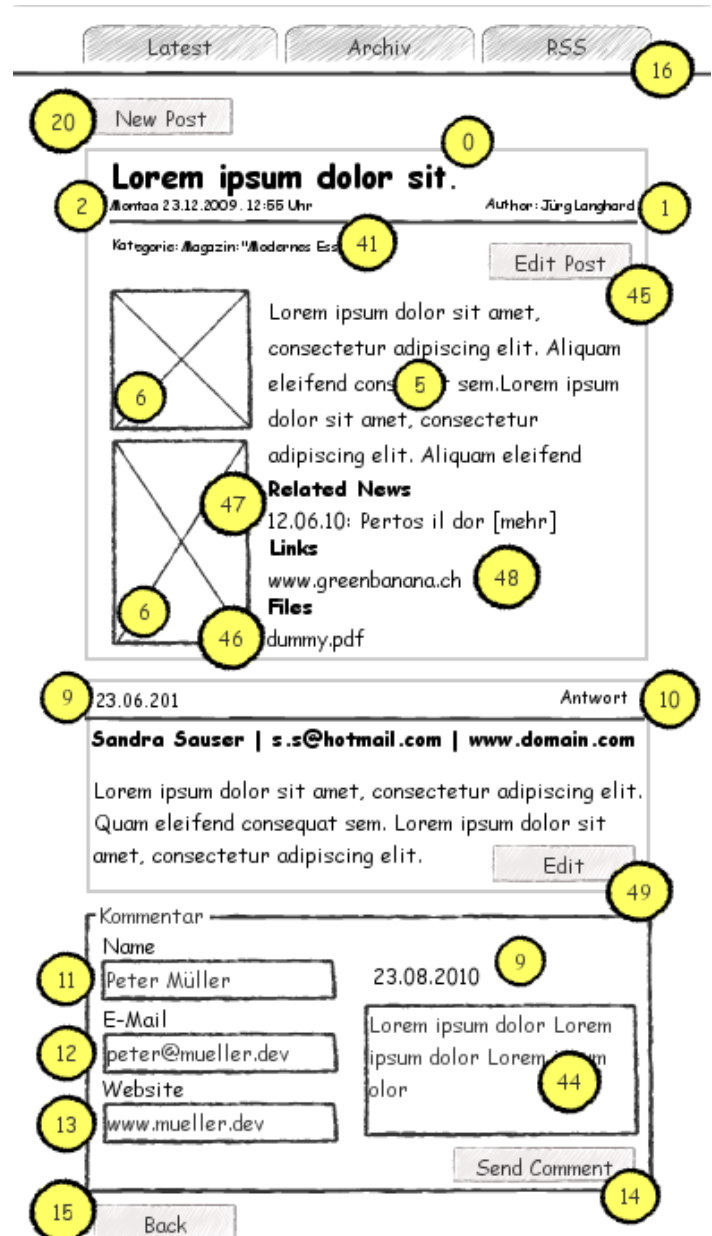
Beim Klicken auf den Navigationsbutton „RSS“ in der News-Navigation (16) wird der Feed aufgerufen.



Post / Single

Generell

Die Single-, oder auch Detailansicht genannt, ist das Herz des News-Systems. Sie zeigt den übergebenen Eintrag in voller Grösse und hat mit dem Kommentar-System zudem einen Interaktiven Part.



Frontend-Editing

Frontend-Editing bedeute, dass sämtliche Einträge, Kategorien und Kommentare über das Frontend editiert werden können. Diese vereinfacht den Benutzer des Systems die Arbeit durch die Möglichkeit des intuitiven Handlings.

Post / Edit (FE-Editing)

Generell

Die Ansicht „Post-Edit“ bietet die Möglichkeit News-Einträge zu erfassen, zu editieren und/oder zu löschen. Da diese Ansicht vor unbefugtem Verwenden geschützt werden muss, kann sie nur aufgerufen werden, wenn Benutzer über die Rechte für das FE-Editing verfügen.

The screenshot shows the 'Post / Edit (FE-Editing)' interface with the following elements and annotations:

- Type:** Dropdown menu set to 'Post' (19).
- Title:** Text input field containing 'Free-Climbing' (0).
- Post-Image:** Section with three image placeholders, each with a 'Delete' button (6, 17, 17).
- Hide by SingleView:** Checkmark button (18).
- Author:** Text input field containing 'Jürg Langhard' (1).
- Date:** Text input field containing '23-09-2010' with a calendar icon (2).
- Category:** List of categories with checkboxes: Sport (3), Wintersport, Free-Climbing, OpenSource, TYPO3, PHP, FLOW3 (22).
- Buttons:** 'Edit Category' (21) and 'New Category' (21).
- Links:** Table with columns 'Type', 'Link', and 'Admin'. Entries include URL, Page, and Mail (30).
- Buttons:** 'New Link' (31), 'New Post' (32), and 'Delete Post' (33).
- New Image:** Text input field containing '/localFolder/categoryImage.png' with an 'Upload' button (6).
- Teaser:** Text input field containing Lorem ipsum text (5).
- Post:** Text input field containing Lorem ipsum text (5).
- Related News:** Table with columns 'Nr 1', 'Date', 'Name', 'Comment', and 'Admin'. Entries include Paul, Peter, and Sandra (27).
- Buttons:** 'New Relation' (28).
- Comments:** Table with columns 'Nr 1', 'Date', 'Name', 'Comment', and 'Admin'. Entries include Paul Kleiner, Peter Müller, and Sandra Poll (26, 49).
- Files:** Table with columns 'Nr 1', 'File', and 'Admin'. Entries include dummy.pdf and Ipsum.doc (25).
- Buttons:** 'New File' (29), 'Cancel' (34), and 'Save' (35).

Comment/Edit (FE-Editing)

Generell

Die Ansicht „Comment/Edit“ kann sowohl über „Post/Single“ also auch über „Post/Edit“ aufgerufen werden.

Kommentar

Name: Peter Müller

E-Mail: peter@mueller.dev

Website: www.mueller.dev

23.08.2010

Comment content: Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor

Buttons: Delete Comment, Cancel, Save

Category/Edit (FE-Editing)

Generell

In dieser Ansicht kann können die News-Kategorien verwaltet werden.

Mutter-Kategorie

☒ Sport

☐ Fussball

☐ Handball

☐ Wintersport

☐ Free-Climbing

☐ OpenSource

☐ TYPO3

☐ PHP

☐ FLOW3

Title: Free-Climbing

Category-Image: /localFolder/categoryImage.png

Upload

Description: Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor Lorem ipsum dolor

Buttons: New Category, Delete Category, Cancel, Save

Positionsbeschreibung

0 - Überschrift

Jeder Eintrag hat eine eigene Überschrift. Diese soll es dem Redakteur ermöglichen den Leser zu fesseln und so zu animiert den Teaser-Text (5) und später die ganze News zu lesen.

Die Überschrift ist verlinkt mit der Detailansicht (Post/Single) des jeweiligen Eintrages.

1 - Autor

Jedem Eintrag kann hinterlegt werden, wer der Autor ist. Die Ausgabe ist rein informativ und hat keine hinterlegten Funktionalitäten.

Spezielles: Wird kein Autor angegeben, erscheint keine Anzeige.

2 - Datum

Das Datum gibt an, wann der Eintrag geschrieben wurde.

Darstellungsform: dd.mm.yyyy hh:mm.

3 - Kategorie-Menü

Dieses Menü zeigt die Kategorien in einer hierarchischen Ansicht. Die Menüpunkte können mittels Checkbox gewählt werden und definieren nach dem Reload (37) die anzuzeigenden News-Einträge.

Bei Mehrfachauswahl wird das Logische AND verwendet.

5 - Teaser-Text

Der Teaser-Text ist dazu da, den News-Eintrag so spannen wie möglich zu umschreiben und den Leser zu animieren den ganzen Eintrag zu lesen. Wird er vom Redakteur nicht explizit angegeben, wird er aus dem News-Text vom System generiert.

6 - Bild

Jedem Eintrag können mehrere Bilder hinterlegt werden. Das erste Bild wird für die Anzeige in dieser Ansicht verwendet und ist, wie die Überschrift, mit der Detailansicht des jeweiligen Eintrages verlinkt.

7 - Link zur Detailansicht (Post/Single)

Dieser Text ist verlinkt mit der Detailansicht des jeweiligen Eintrages. Der Linktext kann per TypoScript definiert werden.

8 - Pagebrowse

Gibt es mehr Einträge in der Latest-Ansicht als angezeigt, werden sollen, erscheint eine Pagebrowse-Navigation.

9 - Kommentar-Datum

Jedem Kommentar wird das Erstellungsdatum hinterlegt.

10 - Antworten

Dieser Text ist frei wählbar und ist verlinkt. Durch Klicken auf den Text wird das Eingabeformular der Kommentarfunktion sichtbar.

11 - Kommentar-Name

Eingabefeld für den Name des Kommentar-Autors. Es handelt sich dabei um ein Pflichtfeld und darf nicht leer gelassen werden.

12 - Kommentar-E-Mail

Der Verfasser des Kommentars hat hier die Möglichkeit eine E-Mail zu hinterlegen, um gegebenenfalls kontaktiert werden zu können. Diese Eingabe ist optional und darf auch leer gelassen werden.

13 - Kommentar-Website

Eingabefeld für die Website des Kommentar-Autors. Diese Eingabe ist ebenfalls optional und darf auch leer gelassen werden.

14 - Button „Send Comment“

Sind alle Felder korrekt ausgefüllt, kann das Formular abgeschickt werden.

Die Kommentare sind dann sofort sichtbar.

15 - Button „Back“

Die ist lediglich ein Navigationsbutton und führt den Leser zurück zur Seite auf der die Latest-Ansicht (Post/Latest= implementiert ist. Welche Seite das ist, kann per TypoScript definiert werden.

16 - News-Navigation

Die News-Navigation ermöglicht ein rasches Navigieren zwischen den Ansichten „RSS“ und „Archiv“ im Frontend.

19 - Type

Es gibt grundsätzlich zwei verschiedenen Typen von News-Einträgen: Post und Link.

Ein "Post" ist einen Eintrag, bei dem die vorgegebenen Formular-Felder für die Contenteingabe genutzt werden.

Ein Link kann verwendet werden, um auf eine interne oder externe Seite zu verweisen. Wird dieser Typ verwendet, sind nur die folgenden Eingabemasken aktiv:

- Title
- News-Image
- Teaser
- Tags
- Autor
- Date
- Link

Bei dieser Anzeige gibt es keine Single-Ansicht. Es wird lediglich einen Eintrag in der Liste von „Post/Latest“ angezeigt.

20 - Button „New Post“ (FE-Editing)

Sobald ein Benutzer die Berechtigung hat das Frontend-Editing zu verwenden, erscheint dieser Button. Er kann verwendet werden, um einen News-Eintrag zu verfassen.

Der Button ist verlinkt mit dem Mockup „Post/Edit“ und erzeugt ein leeres Objekt.

25 - Files

Auflistung von Files, welche dem News-Eintrag angehängt sind. Diese werden nur auf der Detailansicht (Post/Single) angezeigt.

26 - Comments

In dieser Ansicht werden hierarchisch alle Kommentare zum gewählten Post angezeigt. Bearbeiten oder gar Löschen eines Kommentars kann man in der Ansicht „Comment/Edit“ welche man über den Admin-Link (49) erreicht.

27 - Related News

Diese Ansicht zeigt alle News welche verwandt sind mit dem gewählten Post. Weiter Relationen können

über den Button „New Relation“ angehängt werden.

28 - Button „New Relation“

Durch einen Klick auf diesen Button erscheint eine Anzeige mit allen bis anhin erfassten News-Einträge welche als „Related News“ definiert werden können.

29 - Button „New File“

Durch Klicken auf den Button „New File“ erscheint eine Eingabemöglichkeit um Dateien auf den Server zu laden.

30 - Links

Auflistung von Links, die mit dem News-Eintrag in Verbindung stehen.

Um einen Link zu löschen, kann auf den verlinkten Text „remove“ in der Liste geklickt werden.

31 - Button „New Link“

Um einen neuen Link einzufügen, kann auf den Button mit der Aufschrift „New Link“ geklickt werden. Es erscheint dann ein Kontextmenü über welches ein neuer Eintrag erfasst werden kann.

32 - Button „New Post“

Über diesen Button kann ein neuer News-Eintrag erstellt werden. Alle Eingabefeld sind leer (bis auf das Datum, welches mit dem aktuellen Datum vorbelegt ist)

33 - Button „Delete Post“

Ein bereits im System erfasster News-Eintrag kann über diesen Button gelöscht werden.

34 - Button „Cancel“

Wurden an einem Eintrag Änderungen vorgenommen, die man so nicht speichern möchte, kann die Eingabe mit dem Button „Cancel“ abgebrochen werden.

35 - Button „Save“

Änderungen oder einen neuen News-Eintrag kann mit diesem Button gespeichert werden.

37 - Button „New Category“

Mit diesem Button kann eine leere Ansicht von „Category - Edit“ aufgerufen werden, welche das Erstellen einer neuen Kategorie ermöglicht.

38 - Button „Delete Category“

Eine bestehende Kategorie kann mit diesem Button gelöscht werden.

39 - Button „Cancel“

Wurden an einer Kategorien Änderungen vorgenommen, die so nicht gespeichert werden sollen, kann mit diesem Button alles wieder zurückgesetzt werden.

40 - Button „Save“

Neue und geändert Kategorien können mit diesem Button gespeichert werden.

41 - Kategorie

Jeder einzelne Kategorie-Eintrag ist mit der Archiv-Ansicht verlinkt. Die Archiv-Ansicht (Post/Archiv) zeigt dann alle Einträge der angeklickten Kategorie.

42 - Kategorie-Image

Symbolbild der Kategorie. (Wird im Moment noch nicht verwendet im Frontend)

43 - Kategorie-Beschreibung

Beschreibung der Kategorie. (Wird im Moment noch nicht verwendet im Frontend).

44 - Kommentar

In diesem Textfeld kann der Kommentar-Text verfasst werden. Diese Eingabe ist auch ein Pflichtfeld und muss eine bestimmte Zeichenanzahl beinhalten.

45 - Button „Edit Post“ (FE-Editing)

Sobald ein Benutzer die Berechtigung hat das Frontend-Editing zu verwenden, erscheint dieser Button. Er kann verwendet werden, um einen News-Eintrag zu editieren.

Der Button ist verlinkt mit dem Mockup „Post/Edit“.

49 - Button „Edit“ (FE-Editing)

Auch die Kommentare können über das FE-Editing mit den nötigen Berechtigungen editiert oder gar gelöscht werden. Wird der Button „Edit“ angeklickt, erscheint der entsprechende Kommentar in der Ansicht „Comment/Edit“.

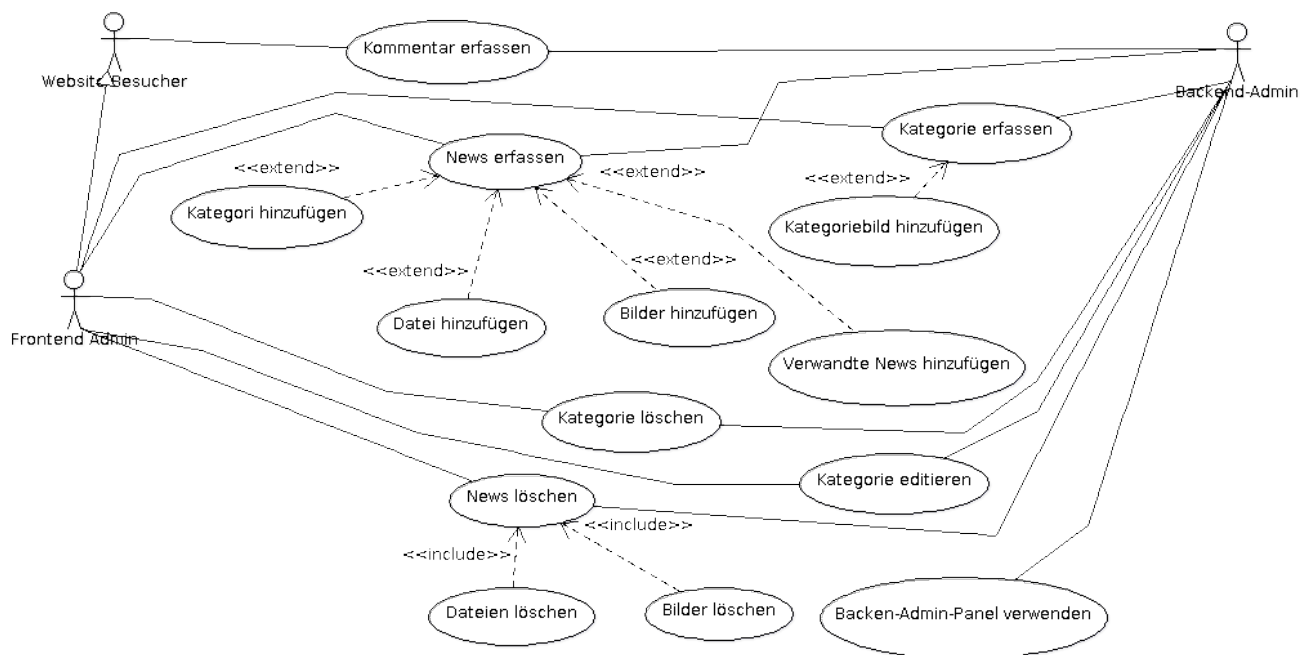
Vom Wireframe zur Extension

Die Frage „Was soll mein geplantes System eigentlich leisten?“ sollte am Beginn jeder Systementwicklung stehen. Eine fundierte Beantwortung dieser Frage bewahrt davor, im Detail zu versinken, bevor man weiss, was vom System überhaupt erwartet wird.

Aus meiner Erfahrung in der Software-Entwicklung weiss ich, dass es oftmals so ist, dass der Kunde lieber anhand Wireframes die Funktionalität besprechen möchte, da diese noch ein Stück weniger abstrakt sind als UML-Diagramme. Für eine abschliessende Besprechung und eine andere Ansicht der Aufgabenstellung ist es aber denn noch sehr nützlich anhand eines UML-Diagramms die Funktionalität mit dem Kunde nochmals durchzusprechen.

UML

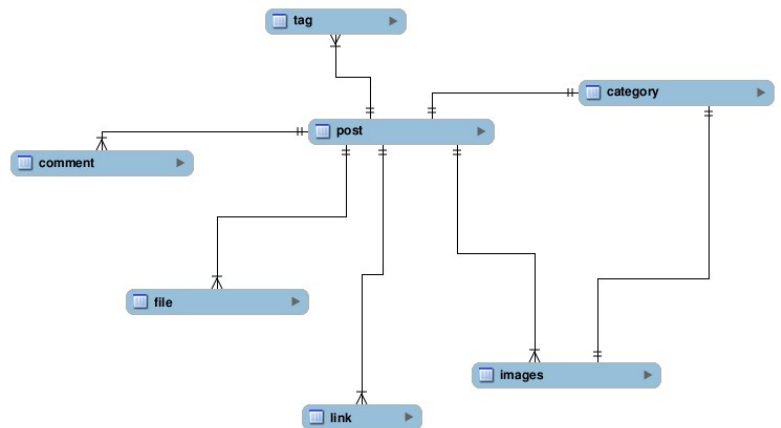
Ein vereinfachtes UML der Extension kiddog_news würde so aussehen:



Datenbankentwurf

Domänenanalyse

Der nächste Schritt ist das Finden der richtigen Domänen. Hierfür müssen die Anforderungen aus den Wireframes in „reale“ Domänen verpackt und anschließend den folgenden Domänen-Arten zugeordnet werden:



Arten des Modells

Entities (Entitäten)

Eine Entität ist ein Domänenobjekt, das eine globale und beständige Identität besitzt. Beispielsweise kann dies eine eindeutige Zuordnung zu einer Person sein. Zur Zuordnung benötigt man eine eindeutige ID oder eine Kombination aus mehreren Feldern. Hat man diesen Schlüssel, so kann man das jeweilige Objekt eindeutig bestimmen.

Dies ist in etwa vergleichbar mit Datenbanken, in denen man einen Datensatz mithilfe einer UID (oder einer Kombination mehrerer Schlüssel) ermitteln kann.

Value (Object)

Dieses Domänenobjekt besitzt im Gegensatz zur Entity keine globale Identität, während die Identität selber aber weiterhin von grossem Interesse ist. In einer exemplarischen User-Verwaltung wären die User-Daten u.U. Entitäten, während die Adressen Value Objects wären. Ändert sich der Adressdatensatz (weil der User z.B. umgezogen ist), so verliert diese Information ihren Wert und wird durch den aktuellen Datensatz ausgetauscht. Zudem hängt der Adresszusatz immer an einer Entität, was im ersten den eigentlichen Wert gibt.

Service

Als Service werden domänenweite Funktionen bezeichnet, die nicht eindeutig einem Objekt zugeordnet werden können. So kann eine Domäne, die für die Transaktion von Geld (von Konto A auf Konto B) zuständig ist, weder dem einen noch dem anderen Konto zugeordnet werden.

Aggregates-Root

Diese Domänenobjekte sind die Wurzeln von einer Gruppierung von abhängigen Domänen. Es ist im weitesten Sinne wie die Vererbung in der Programmierung zu verstehen:

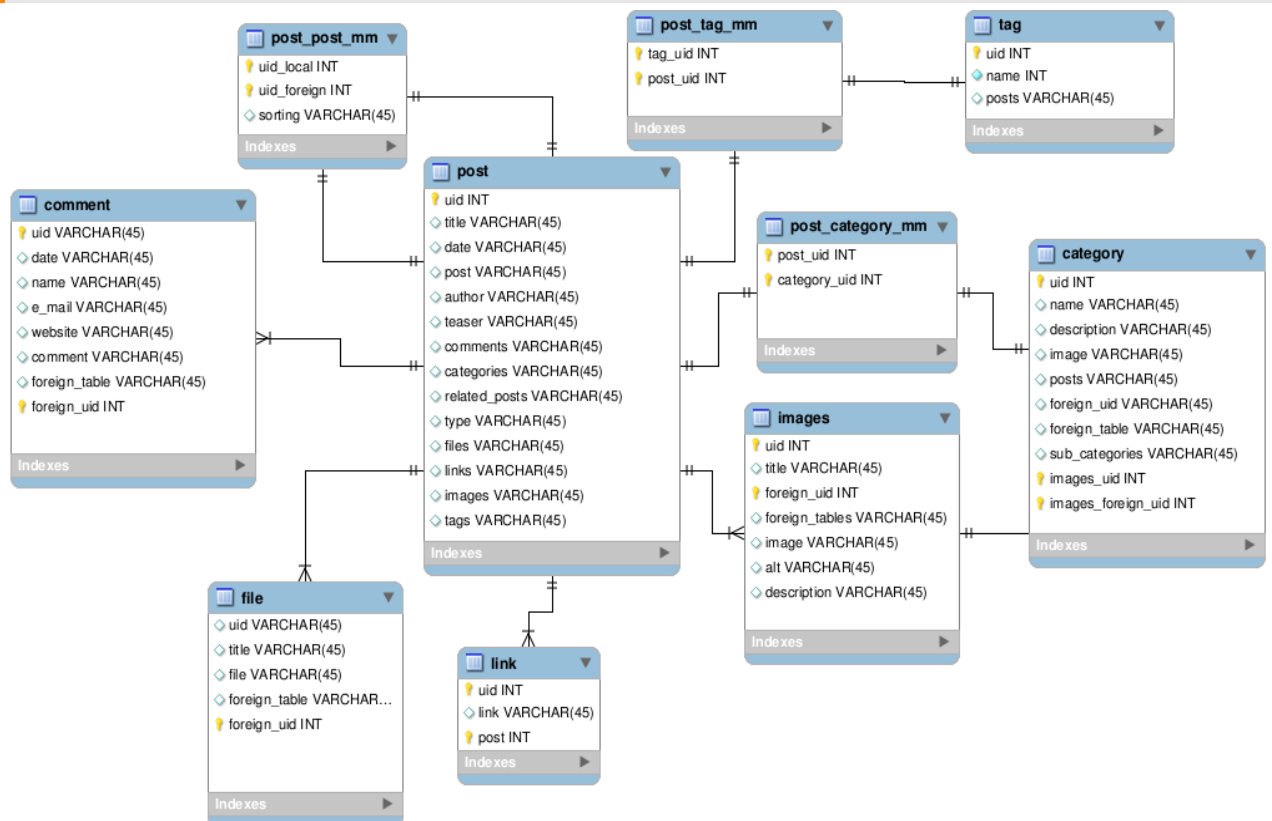
News->Eintrag->Kommentar->Autor

Anpassung des Datenbankentwurfes für Extbase

FLOW3 bietet mit dem Persistenz Framework bereits von Haus aus eine Möglichkeit, Domänenobjekte ohne weitere Konfiguration persistent zu speichern. Bei Extbase hingegen müssen nach wie vor die Datenbanktabellen zusammen mit der Extension angelegt werden. Dazu weist Extbase jeder Klasse des Domänenmodells eine einzelne Datenbanktabelle zu.

Referenzen zwischen Objekten werden in der Datenbank als Fremdschlüsselbeziehungen abgebildet. Dieses Vorgehen nennt sich objektrelationales Mapping.

Entwurf der Tabellen für die neue Extension



Vorgehensweise der Umsetzung

Test-driven Development

Bei herkömmlichen Vorgehensmodellen der Software-Entwicklung mit einer Testphase gegen Projektende bleibt für umfangreiches Testen oft nur wenig Zeit. Vor allem aber können Software-Änderungen, deren Notwendigkeit häufig erst in den Tests erkannt wird, nur mühsam und mit grossem Aufwand noch in den Programmcode integriert werden. Kurz: Die Tests erfolgen viel zu spät.

Test-driven-Development (testgetriebene Entwicklung), eine inzwischen bewährte Methode in der Software-Entwicklung, geht einen anderen Weg. Hier sind der Aufbau von Testfällen und die Durchführung von Tests nicht Endpunkt der Entwicklung, sondern ihr Ausgangspunkt. Hat man die erforderlichen Modelle erstellt, leiten die Tests die Programmierung und nicht umgekehrt. Wie aber soll man Programmcode testen, der noch gar nicht existiert? Frank Westphal, Autor des Buches „Testgetriebene Entwicklung mit JUnit & FIT“, hat das mal so erklärt:

„Testgetriebene Entwicklung geht von einem fehlschlagenden Test aus. Software wird in kleinen sicheren Schritten entwickelt, die abwechselnd darauf abzielen, eine neue Anforderung zu implementieren (den fehlgeschlagenen Test also zu erfüllen) und das Design zu verbessern (und dabei weiterhin alle Tests zu bestehen).“

Die Software-Entwicklung ist demnach ein zyklischer Prozess, „Test – Develop – Test“, in kleinen Schritten: Jede funktionale Programmänderung wird durch einen Test motiviert; jeder neue Test baut auf dem vorhergehenden Test auf, erweitert und verbessert ihn unter Beibehaltung der schon vorhandenen Funktionalität.

Voraussetzung für dieses Vorgehen sind jederzeit wiederholbare, weitestgehend automatisierte Unit Tests – daher auch entsprechende Testwerkzeuge. Verschiedene Tools unterstützen das Test-driven-Development. Sie signalisieren einen erfolgreichen Test durch einen grünen Balken und einen fehlgeschlagenen Test durch einen roten Balken. Entwickler sprechen deshalb von „grünen“ und „roten“ Tests.

Nach jedem erfolgreichen Unit Test wird ein Refactoring durchgeführt. Das heisst, der Code wird restrukturiert und optimiert („aufgeräumt“), um ihn an Code-Richtlinien anzupassen, Redundanzen zu entfernen etc. Ziel ist ein schlichter, übersichtlicher und erweiterungsfähiger Code.

Die automatisierten Unit Tests liefern dem Entwickler ständiges Feedback und zugleich eine ausführbare Spezifikation (die Unit Tests dokumentieren jede neue Funktionalität). Eventuelle Fehler und Mängel werden frühzeitig erkannt und können sofort behoben werden; es werden auch keine Testfälle übersehen. Zusätzlich stellen – automatisierte – System- und Akzeptanztests sicher, dass die aufgebauten Softwarekomponenten als grössere, integrierte Systemeinheiten die Kundenanforderungen exakt erfüllen.

Test-driven-Development und Refactoring bringen dem Kunden Softwarelösungen mit schlanken, ausgetesteten, wartungs-, änderungs- und erweiterungsfreundlichen Anwendungen.

FLOW3, Extbase und Fluid

Grundlagen

Im Vergleich zur Vergangenheit werden heute zunehmend komplexere Geschäftsapplikationen innerhalb von TYPO3 realisiert. Um diesem Anspruch gerecht zu werden, wurde vom Entwicklerteam der TYPO3-Version 5.0 schon recht früh festgelegt, dass diese und zukünftige Versionen auf einem komplett neuen Gerüst aufsetzen sollen.

Dieses neue Gerüst wurde auf den Name FLOW3 getauft und ist ein Enterprise Application Framework, welches auch für beliebig andere – von TYPO3 unabhängige – Anwendungen verwendet werden kann.

Während den Transition Days im Oktober 2008 wurde das sogenannte Berliner Manifest aufgestellt, welches besagt dass TYPO3 v4 und TYPO3 v5 eine gewisse Zeit parallel weiter entwickelt werden. Dies bedeutet zum einen, dass viel verwendete Erweiterungen von v4 nach v5 aber auch Neuerungen von v5 nach v4 portiert werden.

Was ist Extbase?

Extbase ist zunächst eine ganz normale Systemextension die ab der TYPO3-Version 4.3.0 als Alternative zur herkömmlichen Extension-Programmierung mittels Erweiterung der tslib_pibase-Klasse verwendet werden kann. Gerne auch parallel, denn Extbase verträgt sich prima mit klassischen Extension. Extbase führt unter anderem die Konzepte Domain Driven Design und das MVC-Pattern in TYPO3 ein. Damit ist es möglich, sehr elegant auch komplexe Erweiterungen zu schreiben, die sich leicht erweitern lassen. Darüber hinaus, und das wird einer der Hauptgründe für die Verwendung sein, werden sich solche Extension leicht in die zukünftige TYPO3-Version 5.0 migrieren lassen.

Für was wird Fluid verwendet?

Fluid, entwickelt von Core-Developer Sebastian Kurfürst, übernimmt dabei die Aufgabe einer Template-Engine und ist für alle Ein- und Ausgaben zuständig. Auch Fluid ist dabei eine neue Systemextension und kann bei Bedarf gegen eine andere Templating-Engine ausgetauscht werden.

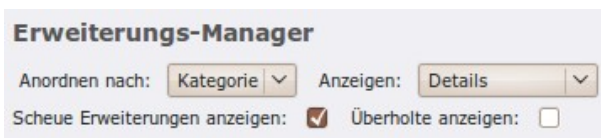
Installation


Um mit Extbase und Fluid arbeiten zu können, wird eine TYPO3-Installation der Version ≥ 4.3 vorausgesetzt.

Nötige Extension installieren

Im Extension-Manager müssen die beiden Systemextension Extbase und Fluid installiert werden. Diese werden zwar mit der TYPO3-Installation mitgeliefert, sind aber per Default noch nicht installiert.



















Damit die Systemextension im Extensions-Manager sichtbar werden, muss die Checkbox „Scheue Erweiterungen anzeigen“ aktiviert werden:



Danach reicht ein einfacher Klick auf das Symbol für die Installation (), und der Installations-Assistent, die mit wenigen, verständlichen Schritte durch die Installation führt, wird angezeigt.

Installation erfolgreich

Wurde alles richtig gemacht, erscheint vor den Extension Extbase und Fluid ein grünes Icon. Dieses symbolisiert, dass die Extension installiert und zur Verwendung bereit ist.

Verschiedenes							
		A Framework for Extensions	extbase	1.0.2		System	Beta
		Static Info Tables	static_info_tables	2.1.1		Lokal	Stabil
		TemplaVoila!	templavoila	1.4.1		Lokal	Stabil
Frontend							
		CSS styled content	css_styled_content	1.0.0		System	Stabil
		Fluid Templating Engine	fluid	1.0.2		System	Beta
		Simulate Static URLs	simulatestatic	1.0.0		System	Stabil

TYPO3-Seite vorbereiten

Damit die Inhalte einer TYPO3-Erweiterung angezeigt werden können, benötigt man ein PAGE-Objekt. innerhalb von TYPO3. Dieses wird mit der TYPO3-eigenen Konfigurationssprache TypoScript gemacht.

Um den Rahmen dieser Arbeit nicht zu sprengen, wurde das Thema „TypoScript“ abgegrenzt und wird nur punktuell erwähnt. Die aktuellste Referenzliste zu TypoScript finden Sie unter:

http://typo3.org/documentation/document-library/references/doc_core_tsref/4.3.0/view

TypoScript Page-Objekt erstellen

```
page = PAGE
page.10 < styles.content.get
```



Das Erstellen des PAGE-Objekt mittels TypoScript hat nicht mit Extbase zu tun. Es wird benötigt um die Ausgabe einer TYPO3-Seite zu steuern.

Neue Seite innerhalb von TYPO3 erstellen

Als Nächstes wird eine TYPO3-Seite benötigt, bei der das Caching ausgeschaltet wird. Dies ist nützlich während der Entwicklung, da so jede Ausgabe neu gerendert (generiert) wird und sich damit das Verhalten der Extension besser nachvollziehen lässt.



Die Info in der Klammer dient als Hinweis, dass diese Seite vom TYPO3-Caching ausgeschlossen wurde und als Gedankenstütze, dass nach der Entwicklung das Einschalten des Caching nicht vergessen geht.

Das Caching lässt sich entweder global per TypoScript auf dem zuvor generierten PAGE-Objekt ...

```
page.conf.no_cache = 1
```

... oder auf einer bestimmten Seite über die Checkbox „Nicht cachen“ deaktivieren:

Seitentitel:

News-System (no cached)

Alias: Ziel: Nicht cachen: ☒ Cache verfällt:

Neue Extension erstellen

Das Erstellen einer Extension die auf Extbase basiert ist im Moment noch relativ aufwendig und mit viel Handarbeit verbunden, da der Kickstarter dafür noch nicht fertiggestellt wurde.

Verzeichnisse und Basis-Dateien

Die Basis für die Extension bildet das Verzeichnis „kiddog_news“ im Verzeichnis „typo3conf/ext/“. Es ist das Root-Verzeichnis der Extension und beinhalten alle Files (PHP, HTML, TXT, JS usw.) welche für den Betrieb der Extension nötig sind.

Die Struktur einer Extension die mit „Extbase“ und „Fluid“ betrieben wird, unterscheidet sich grundlegend von der bisherigen. Aber alles der Reihe nach. Die Bedeutung der einzelnen Ordner und deren Inhalte werden später beschrieben. Im Moment ist es nur wichtig, dass sie existieren.



Achtung: Es ist zwingend notwendig, dass die Gross- und Kleinschreibung gemäss der Abbildung eingehalten wird.

Als Nächstes braucht es folgende 3 Dateien im Extension-Root-Verzeichnis.

File: `ext_emconf.php`

Diese Datei enthält alle wichtigen Eckdaten der Extension. Dazu gehört der Name, eine Beschreibung, Abhängigkeiten/Konflikte zu anderen Extension, den Name des Entwicklers um nur die wichtigsten zu nennen.



Achtung: Sie unterscheidet sich nicht zu derjenigen, welche in herkömmlichen Extension verwendet wurde. Sie kann also aus einer herkömmlich programmierten Erweiterung kopiert und angepasst werden. Infos dazu im Anhang I (Erstellung mittels Kickstarter).






















File: `ext_localconf.php`

Die Datei `ext_localconf.php` beinhaltet Werte für das Array `$TYPO3_CONF_VARS`, insbesondere die zu ladenden Klassen. Diese Datei wird dann aufgerufen, wenn sie gefunden wird, kann also in gewissen Fällen weggelassen werden.

Nicht aber, wenn mit Extbase gearbeitet wird. Dann wird in diesem File der Dispatcher konfiguriert. Dieser überprüft im ersten Array die richtigen Kombinationen von Controller und Actions und im Zweiten, ob diese Kombinationen gecacht werden sollen..



Achtung: Für die Zeit der Entwicklung rät sich alle Controller und den zugehörigen Actions mit dem Parameter `no_cache = 1` zu versehen. So können eventuelle Fehler die durch gecachte Parameter entstehen vermieden werden.

- ▼  **Classes**
 - ▶  **Controller**
- ▼  **Domain**
 - ▶  **Model**
 - ▶  **Repository**
- ▼  **Configuration**
 - ▶  **FlexForms**
 - ▶  **TCA**
 - ▶  **TypoScript**
- ▼  **Resources**
 - ▼  **Private**
 - ▶  **Language**
 - ▶  **Layouts**
 - ▶  **Partials**
 - ▼  **Templates**
 - ▶  **Category**
 - ▶  **Comment**
 - ▶  **Link**
 - ▶  **Post**
- ▼  **Public**
 - ▶  **Icons**

Namenskonventionen

Den Namen einer Extension bezeichnet man in Anlehnung an die Bezeichnung primärer Schlüssel in Datenbanken als Extension-Key. Dieser Name muss für eine Extension, die man veröffentlichen will, einmalig sein.

Für die Registration/Reservation eines Extension-Key's kann man sich auf der offiziellen TYPO3-Website, www.typo3.org, einen Benutzer-Account anlegen, welcher das Registrieren eines solchen erlaubt.

Erlaubt sind dabei Zeichen von a-z (keine Grossbuchstaben) sowie Zahlen von 0-9 und Unterstriche. Einzige Ausnahme für den Beginn der Extension ist „tx“ und „u“. Diese Kürzel würden innerhalb der TYPO3-Core zu Fehlverhalten führen.

Weshalb 'kiddog_news'?

Da es sich eingebürgert hat, dass ein Extension-Key immer mit demselben Kürzel beginnt, wenn sie von derselben Person (oder demselben Unternehmen) programmiert wurde, verwende ich die Kürzel kiddog_* und grb_* für den Beginn meiner Extension.

kiddog_

entstand aus einem Wortspiel mit dem Name „Kiddo“ (meinem Hund) und „Dog“ (Englisch für „Hund“). Dieser Extension-Key verwendete ich in meiner Zeit als Freelancer sowie für private Projekte.

grb_

entstand aus den markantesten Buchstaben aus dem Firmennamen „GreenBanana GmbH“.

Trotz diesem Quasi-Standard mit dem Anfang des Extension-Key's ist es nicht garantiert, dass diese Kürzel nicht von anderen Programmierern verwendet werden. Dies ist jedoch sehr selten der Fall, da die Wertschätzung von Arbeiten aus der Community gegenseitig sehr hoch ist.

Registrieren der Extension innerhalb von TYPO3

Es ist davon auszugehen, dass nachfolgende Registrierungs-Schritte mit der finalen Version des Kickstarters nicht mehr von Hand gemacht werden müssen. Die ganze Prozedur für einmal Schritt für Schritt erstellen zu müssen, trägt jedoch sehr viel für das Verständnis der Abläufe bei.

FE-Plugin

Damit TYPO3 weiss, das es sich um ein Frontend-Plugin handelt, müssen beim Aufruf einer TYPO3-Seite die entsprechenden Klassen geladen werden. Diese kann mit dem Eintrag der folgenden Zeilen in ext_tables.php gewährleistet werden.

```
Tx_Extbase_Utility_Extension::registerPlugin(
    $_EXTKEY,
    'Pil',
    'News-System'
);
```

Backend-Modul

Das selbig gilt für die Registrierung des Backend-Moduls. Damit diese Klassen jedoch nicht unnötigerweise bei jedem Frontend-Aufruf geladen werden, wird mit einer einfachen if-Abfrage überprüft, ob das Backend geladen wurde und somit die Klassen der Erweiterung benötigt.

```
/** *****
 * Backen-Modul
 ***** */
if (TYPO3_MODE === 'BE') {
    /**
     * Registers a Backend Module
     */
    Tx_Extbase_Utility_Extension::registerModule(
        $_EXTKEY,
        'web', // Make module a submodule of 'web'
        'tx_kiddognews_m1', // Submodule key
        ''
    );
}
```

Extension installieren

Die soeben erstellte Erweiterung kann nun, wie schon zu Beginn die Extension „extbase“ und „fluid“, über den Extension-Manager installiert werden.

Durch diese Installation wird der Extension-Key in die `localconf.php` geschrieben, welche verantwortlich ist für das Laden und Bereitstellen der erweiterten Klassen.

Damit nach dem weiter oben besprochenem Prinzip „Test-driven-Development“ entwickelt werden kann, macht die Installation bereits in diesem Stadium der Entwicklung Sinn.

Konfiguration der Datenbanktabellen

Die Struktur der Erweiterung wurde bereits festgelegt und damit auch die zu verwendenden Tabellen und Tabellenfelder.

Ganz nach dem Motto: „Convention over Configuration“ muss die bereits definierte Datenbankstruktur noch den Extbase-Konventionen angepasst werden.

Die Fachmodell-Klassen werden später per Konvention den Namen Tx_KiddogNews_Domain_Model_Project usw. tragen. Eine weitere Konvention besagt, dass die Datenbanktabellen für ein Fachmodell dieselben Namen, in CamelCase-Schreibweise, tragen muss.

Die Domänen-Analyse hat folgendes Ergebnis gegeben:

Domain-Models

Post

• Title	(Text String)	Mockup-Position: 00
• Autor	(Text String)	Mockup-Position: 01
• Date	(Date Time)	Mockup-Position: 02
• Categories	(DomainModel:Category)	Mockup-Position: 41
• Images	(Domain-Model:File)	Mockup-Position: 06
• Teaser	(Text String)	Mockup-Position: 05
• Post	(Text String)	Mockup-Position: 05
• RelatedNews	(Domain Model: Post)	Mockup-Position: 47
• Comments	(Domain Model: Comment)	
• Files	(Domain Model: File)	Mockup-Position: 46
• Links	(Domain Model: Link)	Mockup-Position: 48

Comment

• Date	(Date Time)	Mockup-Position: 09
• Name	(Text String)	Mockup-Position: 11
• E-Mail	(Text String)	Mockup-Position: 12
• Website	(Text String)	Mockup-Position: 13
• Comment	(Text String)	Mockup-Position: 44

Category

- | | | |
|-------------------|--------------------------|---------------------|
| • Title | (Text String) | Mockup-Position: 41 |
| • Image | (Text String) | Mockup-Position: 42 |
| • Description | (Text String) | Mockup-Position: 43 |
| • Parent Category | (Domain Model: Category) | Mockup-Position: 03 |

File

- | | |
|---------------|---------------|
| • File | (Text String) |
| • Description | (Text String) |

Link

- | | |
|--------|---------------|
| • Type | (Text String) |
| • Link | (Text String) |

Tag

- | | |
|--------|---------------|
| • Name | (Text String) |
|--------|---------------|

Datenbankstruktur

Abgeleitet aus den Domain-Models erhält man folgende Datenbanktabellen:

- tx_kiddognews_domain_model_post
- tx_kiddognews_domain_model_comment
- tx_kiddognews_domain_model_category
- tx_kiddognews_domain_model_file
- tx_kiddognews_domain_model_link
- tx_kiddognews_domain_model_tag

File: ext_tables.sql

Die Datei ext_tables.sql liegt im Extension-Root und beinhaltet den DB-Dump, welcher für den Betrieb der jeweiligen Extension eingespielt werden muss. Dies passiert automatisch bei der Installation mittels Extension-Manager.

Werden Änderungen an dieser Datei vorgenommen, nachdem die Erweiterung bereits installiert ist, können diese auch ohne direkten Eingriff in die Datenbank, über den Extension-Manager ausgeführt werden.

Für die Erweiterung kiddog_news sieht diese Datei (vereinfacht) folgendermassen aus:

Beispiel: ext_tables.sql von kiddog_news

```
CREATE TABLE tx_kiddognews_domain_model_post (
    uid int(11) unsigned DEFAULT '0' NOT NULL auto_increment,

    title tinytext,
    date int(11) DEFAULT '0' NOT NULL,
    post tinytext,
    author tinytext,
    teaser tinytext,
    post text NOT NULL,
    comments int(11) unsigned DEFAULT '0'
    categories int(11) unsigned DEFAULT '0'
    tags int(11) unsigned DEFAULT '0' NOT NULL,
    related_posts int(11) unsigned DEFAULT '0'
    type int(11) unsigned DEFAULT '0'
    files int(11) unsigned DEFAULT '0'
    links int(11) unsigned DEFAULT '0'
    images int(11) unsigned DEFAULT '0'
    tstamp int(11) unsigned DEFAULT '0' NOT NULL,
    crdate int(11) unsigned DEFAULT '0' NOT NULL,
    deleted tinyint(4) unsigned DEFAULT '0' NOT NULL,
    hidden tinyint(4) unsigned DEFAULT '0' NOT NULL,

    PRIMARY KEY (uid),
    KEY parent (pid),
);

CREATE TABLE tx_kiddognews_domain_model_comment (
    uid int(11) unsigned DEFAULT '0' NOT NULL auto_increment,

    foreign_uid varchar(30) DEFAULT '' NOT NULL,
    foreign_table tinytext NOT NULL,

    date tinytext,
    name tinytext,
    e_mail tinytext,
    website tinytext,
    comment text NOT NULL,

    PRIMARY KEY (uid),
    KEY parent (pid),
);

CREATE TABLE tx_kiddognews_domain_model_category (
    uid int(11) unsigned DEFAULT '0' NOT NULL auto_increment,

    foreign_uid varchar(30) DEFAULT '' NOT NULL,
    foreign_table tinytext NOT NULL,

    name varchar(255) DEFAULT '' NOT NULL,
    posts int(11) unsigned DEFAULT '0' NOT NULL,
    description tinytext,
    sub_categories int(11) unsigned DEFAULT '0',
    image int(11) unsigned DEFAULT '0',

    PRIMARY KEY (uid),
    KEY parent (pid),
);

CREATE TABLE tx_kiddognews_domain_model_tag (
    uid int(11) unsigned DEFAULT '0' NOT NULL auto_increment,

    name varchar(255) DEFAULT '' NOT NULL,
    posts int(11) unsigned DEFAULT '0' NOT NULL,

    PRIMARY KEY (uid),
    KEY parent (pid),
);
```

```
CREATE TABLE tx_kiddognews_post_tag_mm (
    uid_local int(11) unsigned DEFAULT '0' NOT NULL,
    uid_foreign int(11) unsigned DEFAULT '0' NOT NULL,
    sorting int(11) unsigned DEFAULT '0' NOT NULL,
    sorting_foreign int(11) unsigned DEFAULT '0' NOT NULL,
    KEY uid_local (uid_local),
    KEY uid_foreign (uid_foreign)
);

CREATE TABLE tx_kiddognews_domain_model_file (
    uid int(11) unsigned DEFAULT '0' NOT NULL auto_increment,
    pid int(11) DEFAULT '0' NOT NULL,
    title varchar(255) DEFAULT '' NOT NULL,
    file text NOT NULL,
    foreign_uid varchar(30) DEFAULT '' NOT NULL,
    foreign_table tinytext NOT NULL,
    tstamp int(11) unsigned DEFAULT '0' NOT NULL,
    crdate int(11) unsigned DEFAULT '0' NOT NULL,
    deleted tinyint(4) unsigned DEFAULT '0' NOT NULL,
    hidden tinyint(4) unsigned DEFAULT '0' NOT NULL,
    PRIMARY KEY (uid),
    KEY parent (pid),
);

CREATE TABLE tx_kiddognews_domain_model_image (
    uid int(11) unsigned DEFAULT '0' NOT NULL auto_increment,
    alt varchar(255) DEFAULT '' NOT NULL,
    description varchar(255) DEFAULT '' NOT NULL,
    title varchar(255) DEFAULT '' NOT NULL,
    image text NOT NULL,
    foreign_uid varchar(30) DEFAULT '' NOT NULL,
    foreign_table tinytext NOT NULL,
    PRIMARY KEY (uid),
    KEY parent (pid),
);

CREATE TABLE tx_kiddognews_domain_model_link (
    uid int(11) unsigned DEFAULT '0' NOT NULL auto_increment,
    post int(11) unsigned DEFAULT '0' NOT NULL,
    link tinytext,
    PRIMARY KEY (uid),
    KEY parent (pid),
);

CREATE TABLE tx_kiddognews_post_category_mm (
    uid_local int(11) unsigned DEFAULT '0' NOT NULL,
    uid_foreign int(11) unsigned DEFAULT '0' NOT NULL,
    tablenames varchar(255) DEFAULT '' NOT NULL,
    sorting int(11) unsigned DEFAULT '0' NOT NULL,
    sorting_foreign int(11) unsigned DEFAULT '0' NOT NULL,
    KEY uid_local (uid_local),
    KEY uid_foreign (uid_foreign)
);

CREATE TABLE tx_kiddognews_post_post_mm (
    uid_local int(11) unsigned DEFAULT '0' NOT NULL,
    uid_foreign int(11) unsigned DEFAULT '0' NOT NULL,
    tablenames varchar(255) DEFAULT '' NOT NULL,
    sorting int(11) unsigned DEFAULT '0' NOT NULL,
    sorting_foreign int(11) unsigned DEFAULT '0' NOT NULL,
    KEY uid_local (uid_local),
    KEY uid_foreign (uid_foreign)
);
```

TCA-Anpassen

Als nächstes muss TYPO3 von den neuen Tabellen in Kenntnis gesetzt werden. Dies geschieht durch eine entsprechende Konfiguration im sogenannten „Table Configuration Array“, kurz TCA welches sich in der Dateien `ext_tables.php` (Befindet sich im Extension-Root-Verzeichnis) und `tca.php` (Befindet sich im Verzeichnis `Configuration/TCA/tca.php`).

Mit der Einführung von Extbase wird das TCA nicht mehr nur für die Darstellung von Formularen im TYPO3-Backend, sondern auch für das objektrelationale Mapping auf die Domainobjekte verwendet.

Eine korrekte und saubere Konfiguration dieses Array ist somit unerlässlich geworden.

Um den Rahmen dieser Semesterarbeit nicht zu sprengen, grenze ich an dieser Stelle die komplette Definition der Möglichkeiten auf die wichtigsten TCA-Konfigurationen ein.

Die komplette Dokumentation zu TCA kann dem Core-Dokument entnommen werden:
http://typo3.org/documentation/document-library/core-documentation/doc_core_api/

Auswahl (TYPE: „select“)

Damit im Backend eine benutzerfreundliche Auswahl getätigt werden kann, ist die Verwendung eines HTML-Select zu empfehlen.

In der Extension `kiddog_news` wird diese Möglichkeit verwendet um den Link-Type zu wählen.

Momentan stehen folgende Möglichkeiten zur Auswahl: Page (Ein Link auf eine Seite, die sich in derselben TYPO3-Installation befindet), External URL (Ein Link auf eine Seite im World Wide Web) sowie die Verlinkung einer E-Mail-Adresse.



tca.php

Bei folgendem Beispiel ist noch anzufügen, dass für die Dokumentation die Sprachdefinitionen (für die mehrsprachige Verwendung der Extension) entfernt wurde. Damit diese Funktionalität gewährleistet ist, müsste anstelle von 'Page', 'External URL' und 'Mail' das entsprechende XML eingebunden werden.

```
'type' => array(
    'exclude' => 0,
    'label'   => 'LLL:EXT:kiddog_news/...:tx_kiddognews_domain_model_link.type',
    'config'  => array(
        'type' => 'select',
        'items' => array(
            array('Page', 1),
            array('External URL', 2),
            array('Mail', 3)
        ),
        'size' => 1,
        'eval' => 'int,required'
    )
),
```


Datenbank

In der Datenbank wird dann der Wert aus dem Array `$(type)[config][item]` gespeichert:

+ Optionen

			uid	pid	post	type	link	tstamp	crdate
<input type="checkbox"/>			1	6	0	2	www.greenbanana.ch	1261558202	1261553820
<input type="checkbox"/>			2	6	0	1	4	1261558225	1261553828
<input type="checkbox"/>			3	6	0	3	info@greenbanana.ch	1261558240	1261553837

Alle auswählen / Auswahl entfernen markierte:

TYPO3-Core-API

Die ausführliche Definition zu „select“ im `$TCA-Array` in der Dokumentation der TYPO3-Core-API:

http://typo3.org/documentation/document-library/core-documentation/doc_core_api/4.2.0/view/4/2/#id4272080

File-Upload (TYPE: „group“, „file“)

Für die einfache Verwendung von Files im TYPO3-Backend gibt es den Type „group“ im `$TCA-Config`. Dieser Type gibt die Möglichkeit Datensätze von unterschiedlichen Tabellen im System zu referenzieren.

Der Type „group“ ist auch das Element, mit welchem Dateien an ein Inhaltselement gebunden werden kann.

Es wird in der Extension „kiddog_news“ für die Administration der Dateien im Domain-Models „File“ verwendet und sieht im TYPO3-Backend folgendermassen aus:

File [1] - TestFile 01

Title

New GreenBanana-Logo

File

logo.png

* -PHP -PHP3

Durchsuchen...

tca.php

```
'file' => array (
    'exclude' => 0,
    'label' => 'LLL:EXT:kiddog_news/:tx_kiddognews_domain_model_file.file',
    'config' => array (
        'type' => 'group',
        'internal_type' => 'file',
        'allowed' => '',
        'disallowed' => 'php,php3',
        'max_size' => $GLOBALS['TYPO3_CONF_VARS']['BE']['maxFileSize'],
        'uploadfolder' => 'uploads/tx_kiddognews',
        'show_thumbs' => 1,
        'size' => 1,
        'minitems' => 0,
        'maxitems' => 1,
    )
),
```

Text-Eingabefeld mit RTE

Auch kann im TCA-Array definiert werden, wenn ein Textfeld mittels einem RTE gefüllt werden soll:

tca.php

```
'description' => array (
    'exclude' => 1,
    'label' => 'LLL:EXT:kiddog_news/:tx_kiddognews_domain_model_category.description',
    'config' => array (
        'type' => 'text',
        'cols' => '30',
        'rows' => '5',
        'wizards' => array(
            '_PADDING' => 2,
            'RTE' => array(
                'notNewRecords' => 1,
                'RTEonly' => 1,
                'type' => 'script',
                'title' => 'Full screen Rich Text Editing|Formatteret redigering i hele vinduet',
                'icon' => 'wizard_rte2.gif',
                'script' => 'wizard_rte.php',
            ),
        ),
    ),
),
```

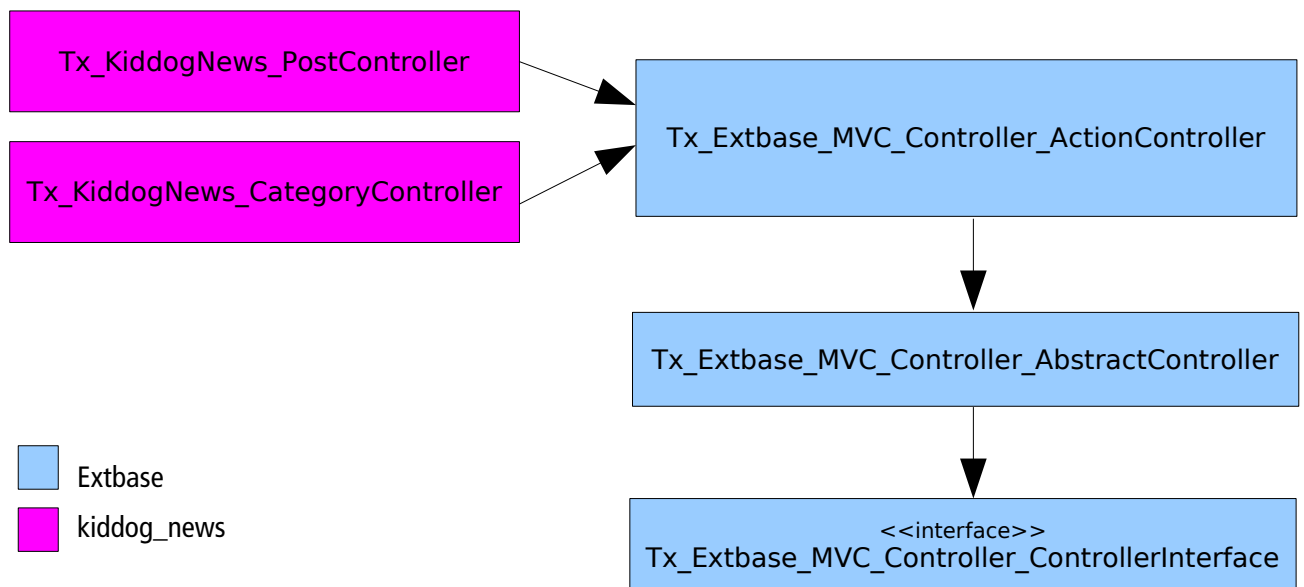
Controller

Die Funktionalität des Controllers wird nun exemplarisch anhand des Post-Controller erklärt. Dafür muss im Verzeichnis Classes/Controller die Datei PostController.php erstellt werden.

Controller in Extbase

Eine weitere Konvention von Extbase ist, dass diese Datei mit „Controller“ endet.

Vererbungsdiagramm der Controller-Klassen



Beispiel-Controller

Dieser Auszug aus dem Post-Controller ist zuständig für die Darstellung der „Latest“-Ansicht und soll dazu dienen die Arbeit des Controllers zu erläutern.

Per Konvention wird jede Aktion als öffentliche Methode implementiert, die dem Namensschema **[Aktionsname]Action** folgt.

```
class Tx_KiddogNews_Controller_PostController
extends Tx_Extbase_MVC_Controller_ActionController {

    /**
     * @var Tx_KiddogNews_Domain_Repository_PostRepository
     */
    protected $postRepository;

    /**
     * Initializes the current action
     *
     * @return void
     */
    public function initializeAction() {
        $this->postRepository =
            t3lib_div::makeInstance('Tx_KiddogNews_Domain_Repository_PostRepository');
    }

    /**
     * showLatest action
     * @return string The rendered latest action
     */
    public function showLatestAction() {
        $this->view->assign('posts',
            $this->postRepository->findLatest($this->settings['time']['archiv']));
    }
}
```



Die Methode **initializeAction()** wird vor jeder anderen Action des Controllers aufgerufen. Auf diese Weise steht in jeder Aktion bereits eine Instanz des Projekt-Repository zur Verfügung, welches für die Interaktion mit der Datenbank verantwortlich ist.

Übergabe von Parameter(n) an einen Controller

Im obigen Beispiel wurde die Action `showLatest()` ohne einen Parameter aufgerufen. Das war auch nicht nötig, da diese Action ja die Aufgabe hat alle News-Post in hierarchischer Reihenfolge darzustellen.

Ganz anders sieht das aus, wenn wir dann eine Einzelansicht anzeigen möchten. Damit das machbar ist, müssen wir zum einen eine Action „showSingle“ im Post-Controller implementiert haben und zum anderen benötigen wir eine eindeutige Kennung, welcher Post angezeigt werden soll.

Mit Extbase können Actions sowohl einzelne Werte wie auch komplette Domänenobjekte als Parameter entgegen nehmen.



Die Verwendung von Doccomment-Kommentaren ist sehr wichtig. Parameter, die nicht im Kommentar dokumentiert sind, oder einen falschen Datentyp aufweisen, quittiert Extbase gnadenlos mit einer Fehlermeldung. Weiter ist die Benennung der Parameter sehr wichtig. So muss der Methodenparameter genau den selben Name tragen wie der GET- oder POST-Parameter, damit eine Zuordnung erfolgen kann.

Beispiel: showSingleAction()

Dieses Beispiel zeigt die `showSingleAction()`, welche für die Darstellung eines einzelnen News-Post verantwortlich ist:

```
/**
 * Action that displays one single post
 * @param Tx_KiddogNews_Domain_Model_Post $post The post to display
 * @return string The rendered view
 */
public function showSingleAction(Tx_KiddogNews_Domain_Model_Post $post) {
    $this->view->assign('post', $post);
}
```

Übergeben wird der eindeutige Post „Tx_KiddogNews_Domain_Model_Post“ welcher angezeigt werden soll. Es handelt sich dabei um das ganze entsprechende Domain-Modell.

Mit der Funktion `$this->view->assign()` wird das Domain-Modell wieder dem „View“ übergeben für die Darstellung.

Verwendete Controller

Im Projekt kiddog_news werden folgende Controller mit den jeweiligen Action verwendet:

PostController

Folgende Actions sind im PostController definiert:

- `showLatestAction()`
- `showArchivAction()`
- `showRssAction()`
- `editAction()`
- `saveAction()`
- `deleteAction()`

CommentController

- `editAction()`
- `deleteAction()`
- `saveAction()`

FileController

- `newAction()`
- `deleteAction()`
- `saveAction()`

CategoryController

- `newAction()`
- `deleteAction()`
- `saveAction()`

LinkController

- `newAction()`
- `deleteAction()`
- `saveAction()`

View

Für die Anzeige von Inhalten sind die „Views“ verantwortlich. Standardmässig ist jeder Controller-Action genau ein View zugeordnet. Die HTML-Vorlage für diese Views werden unter dem Dateiname-Konstrukt „Resources/Private/Templates/[Controllername]/[Actionname].html“ gesucht und von da durch FLUID interpretiert.

Für die Action „showLatest()“ im Post-Controller, muss zum Beispiel folgende Datei erreichbar sein: Resources/Private/Templates/Post/showlatest.html



Wie aus dem obigen Beispiel zu entnehmen ist, kann Extbase und FLUID die CamelCase-Schreibweise beim Dateiname des Templates noch nicht interpretieren. Somit wird aus der Action „showLatest“ das Template „showlatest.html“.

Beispiel View (showlatest.html)

```
<f:layout name="default" />
<f:section name="content">
  <div class="tx_kiddognews-post-showlatest">
    <ul>
      <f:for each="{posts}" as="post">
        <li>
          <f:render partial="postFeEditingMenu" arguments="{post:post}" />
          <div class="tx_kiddognews-post-showlatest-post">
            <f:link.action arguments="{post:post}" action="showSingle">
              <h2>{post.title}</h2>
            </f:link.action>
            <f:format.date>{post.date}</f:format.date>
            <f:if condition="{post.teaser}">
              <f:then>
                <p>
                  {post.teaser}
                </p>
              </f:then>
              <f:else>
                <p>
                  <f:format.crop maxCharacters="100">{post.post}</f:format.crop>
                </p>
              </f:else>
            </f:if>
            <f:link.action arguments="{post:post}"
              action="showSingle"
              controller="Post">
              show
            </f:link.action>
          </div>
        </li>
      </f:for>
    </ul>
  </div>
</f:section>
```

- Integer ac matti**
 2010-05-16
 Integer ac mattis nunc. Praesent vitae purus ut libero consequat feugiat! Cras gravida vehicula...
[show](#)
- Cras in ante sed**
 2010-05-16
 Cras in ante sed , nisl sit amet vulputate tempus, lacus erat euismod augue, sed lacinia turpis...
[show](#)
- Proin neque massa**
 2010-05-16
 Proin neque massa, dignissim id posuere quis, luctus ut velit. Vestibulum ante ipsum primis in...
[show](#)

Mit minimalen CSS-Anweisungen sieht der obige View folgendermassen aus:

Validierung von Eingaben

Extbase stellt auch für das Validieren von Benutzereingaben Methoden bereit, die den Entwicklern jede Menge Arbeit abnehmen.

Möchten man ein einzelnes Attribut eines Domänenobjektes validieren, so kann das über eine einfache Anmerkung in dem DocComment-Kommentar des entsprechenden Attributes realisiert werden.

Beispiel: Attribut „Title“ des Domain-Mode Post

Das Eingabeformular für die Erstellung eines neuen Posts könnte wie das Nebenstehende aussehen.

Der Auszug aus dem Domain-Modell „Post“ sieht folgendermassen aus:

```
class Tx_KiddogNews_Domain_Model_Post
    extends Tx_Extbase_DomainObject_AbstractEntity {

    /**
     * Title
     * @var string
     * @validate StringLength(minimum = 3, maximum = 255)
     */
    protected $title;

    /**
     * Getter for Title
     *
     * @return string Title
     */
    public function getTitle() {
        return $this->title;
    }

    /**
     * Setter for Title
     *
     * @param string $Title Title
     * @return void
     */
    public function setTitle($title) {
        $this->title = $title;
    }
}
```

Create a new Post

Type

Author

Date

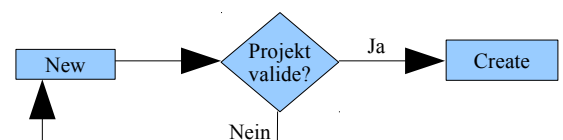
Title (required)

Content (required)

Durch den DocComment-Kommentar „@validate“ kann dem Attribut eine Validierungsregel mitgegeben werden. In diesem Beispiel wird überprüft, ob der String eine Mindestlänge von 3 Zeichen und eine Maximallänge von 255 Zeichen hat.

Validierung = false

Wenn nun diesem Attribut ein neuer Wert zugewiesen wird, wie zum Beispiel bei der Verwendung der Methode `setTitle($title)`, wird der neue Wert automatisch anhand dieser Regel validiert.



Wird nun ein nicht valides Objekt (also zum Beispiel ein Post ohne welcher lediglich ein „X“ als Titel hat, wie auf dem Screenshot zu sehen ist) als Parameter an eine Controller-Aktion übergeben, so leitet uns der Controller automatisch wieder zu der Aktion zurück, von der wir gekommen sind.

Validationsmöglichkeiten in Extbase

Bis jetzt stehe folgende Validationsmöglichkeiten zur Verfügung:

Alphanumerik

Überprüft, ob eine Zeichenkette nur aus Buchstaben (Gross- und Kleinschreibung) und Zahlen besteht.

@validate Alphanumerik

EmailAddress

Überprüft, ob eine Zeichenkette eine E-Mail-Adresse ist.

@validate EmailAddress

NotEmpty

Überprüft, ob ein Attribut entweder einer leere Zeichenkette oder NULL ist.

@validate NotEmpty

NumberRange

Ermittelt, ob ein Attribut einen numerischen Wert enthält und innerhalb eines bestimmten Intervalls befindet.

@validate NumberRange (startRange = 10, endRange = 100)

RegularExpression

Überprüft den Inhalt eines Attributs gegen einen regulären Ausdruck. Der Ausdruck muss vollständig mit Begrenzern angegeben werden.

@validate RegularExpression (regularExpression = '/(Hello| Goodbye) World!/i')

StringLength

Ermittelt, ob sich die Länge einer Zeichenkette innerhalb eines bestimmten Intervalls befindet.

@validate StringLength (minimum=10, maximum=100)

Domain/Repository

Der Datenzugriff auf die Domäne sollte immer, so weit es geht, gekapselt werden. Damit ist ein technologie-neutraler Zugriff auf die Wurzelobjekte möglich, sei es nun via Datenbank oder mittels WebDAV bzw. SOAP. Das Repository liefert die Objekte und kümmert sich aber auch um die Änderung, Versionierung, Speicherung oder Ähnliches, ohne dass dies explizit in Auftrag gegeben werden müsste.

Somit hat letztlich nur das Repository direkt mit der Speicherung der Daten zu tun und letztlich auch mit der Suche nach Objekten im System. Sobald ein angefordertes Objekt gefunden wurde, wird es durch das Repository zurückgeliefert.

Funktionalität

Extbase stellt hierfür die abstrakte Basisklasse `Tx_Extbase_Persistence_Repository` zur Verfügung. Die Verwendung dieser nimmt dem Entwickler jede Menge Arbeit ab.

Dank der Vererbung dieser Klasse stehen den Repository-Klassen bereits folgende gängigen Funktionen zur Verfügung:

Auslesen von Objekten

findAll()

Diese Methode liefert alle Objekte, die nicht gelöscht sind.

findBy[Attribut](\$v)

Liefert alle Objekte, bei denen das durch [Attribut] angegebene Attribut mit \$v übereinstimmt. So könnten mit folgendem Beispiel etwa alle Projekte mit einem bestimmten Namen ermittelt werden:

```
$projects = $repository->findByName("HSZ-T");
foreach($projects as $project){
    echo $project->getName().'<br />';
};
```

findById(\$uid)

Liefert ein einziges Objekt mit einer bestimmten UID.

findOneBy[Attribut] (\$v)

Liefert das erstbeste Objekt, bei dem die durch [Attribut] angegebene Eigenschaft mit \$v übereinstimmt.

Beispiel:

```
$project = $repository->findOneByName("Bar");
If($project == NULL) echo "Nicht gefunden";
Else Echo $project->getName();
```

Manipulieren von Objekten

Das Repository dient nicht nur dazu, Objekte aus der Datenbank auszulesen, sondern auch genauso dazu, Objekte wieder in der Datenbank zu speichern. Hierzu stellt jedes Repository die Methoden add, remove und update zur Verfügung:

add (\$o)

Fügt das Objekt \$o als neues Element in die Datenbank ein. Das Objekt darf als solches noch nicht persistent gespeichert worden sein. Ist das neue Objekt die Wurzel eines Aggregates, so werden die anderen Bestandteile des Aggregates – soweit vorhanden – ebenfalls persistiert.

update (\$o)

Speichert Änderungen, die an dem Objekt vorgenommen wurden, in der Datenbank. Hierzu muss das Objekt bereits existieren.

remove (\$o)

Entfernt ein Objekt wieder aus der Datenbank. Handelt es sich bei dem Objekt um eine Aggregatwurzel, so werden die untergeordneten Bestandteile des Aggregates ebenfalls gelöscht.

Beispiele

```
$postRepository = t3lib_div::makeInstance('Tx_KiddogNews_Domain_Repository_PostRepository');

// Speichern eines neuen Objektes
$postRepository->add($post);

// Bearbeiten eines Objektes
$postRepository->update($post);

// Löschen eines Objektes
$postRepository->remove($post);
```

Erweitern der Repository-Klasse

Alle Funktionen, die bis jetzt beschrieben wurden, stehen dem Entwickler zur Verfügung, wenn er die Abstrakte Klasse Tx_Extbase_Persistence_Repository im jeweiligen Repository einbindet. Es liegt aber auf der Hand, dass diese Funktionen zwar eine grosse Hilfe sind, aber noch nicht reichen, um die Interaktionen einer komplexen Applikation abzubilden.

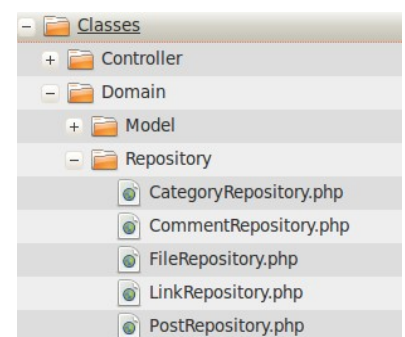
Dafür kann das Query Object Model der jeweiligen Repository-Klasse erweitert werden.

Verwendete Repositories

Die Repository-Klassen werden im Extensions-Verzeichnis Classes/Domain/Repository gespeichert und müssen per Konvention einen Klassennamen haben der mit „Repository“ enden.

Unsere Klassen

- Tx_KiddogNews_Domain_Repository_CategorRepository
- Tx_KiddogNews_Domain_Repository_FileRepository
- Tx_KiddogNews_Domain_Repository_LinkRepository
- Tx_KiddogNews_Domain_Repository_PostRepository
- Tx_KiddogNews_Domain_Repository_CommentRepository



Schlusswort

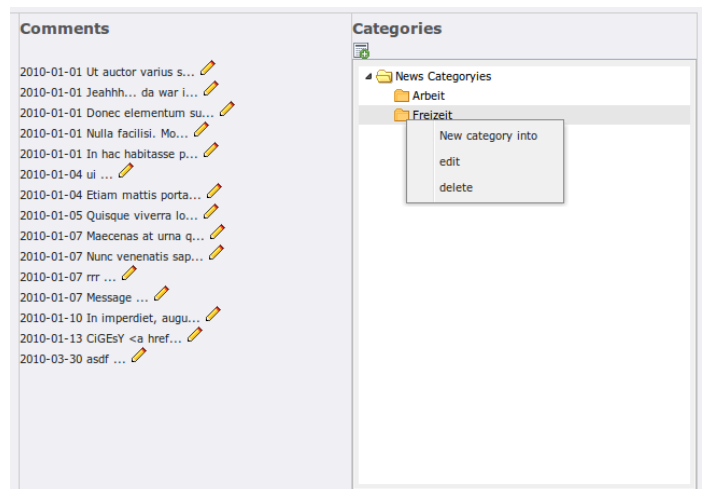
Die Extension "kiddog_news", welche parallel zu dieser Dokumentation von mir programmiert wurde, ist im offiziellen TYPO3-Subversion-Repository einzusehen und wurde auf der Website www.myweblog.ch zu Test- und Präsentationszwecken installiert. Sobald sie den Prototype-Status hinter sich gelassen hat, wird sie selbstverständlich im offiziellen TER (www.typo3.org/ext) für den generellen Download bereitgestellt.

Weiterführende Informationen

Zu erwähnen ist zudem, dass der entwickelte Prototype weitere Funktionalitäten implementiert hat, welche in dieser Semesterarbeit nicht explizit beschrieben wurden. Einen genaueren Blick in den Quellcode lohnt sich also.

So wurde zum Beispiel für die Darstellung des Kategorie-Baumes im Backend eine auf dem Javascript-Framework ExtJS basiertere Baum-Ansicht implementiert, welcher mittels AJAX die nötigen Informationen holt.

Eine Kurzfassung der Verwendung von AJAX innerhalb von Extbase-Basierten TYPO3-Modulen kann im Anhang II nachgelesen werden.



Ziel

Ich hoffe, dass ich dem einen oder anderen mit dieser Dokumentation den Einstieg in die TYPO3-Extension-Programmierung vereinfachen konnte. Über Feedbacks auf info@greenbanana.ch würde ich mich sehr freuen.

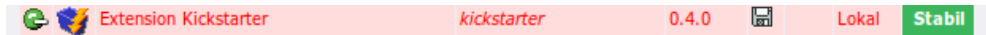
Freundliche Grüsse
Jürg Langhard

Arbeit im Sekretariat der HSZ-T abgegeben am 01. Juni 2010.

Anhang I: Herkömmliche Extension

Erstellung mittels Kickstarter

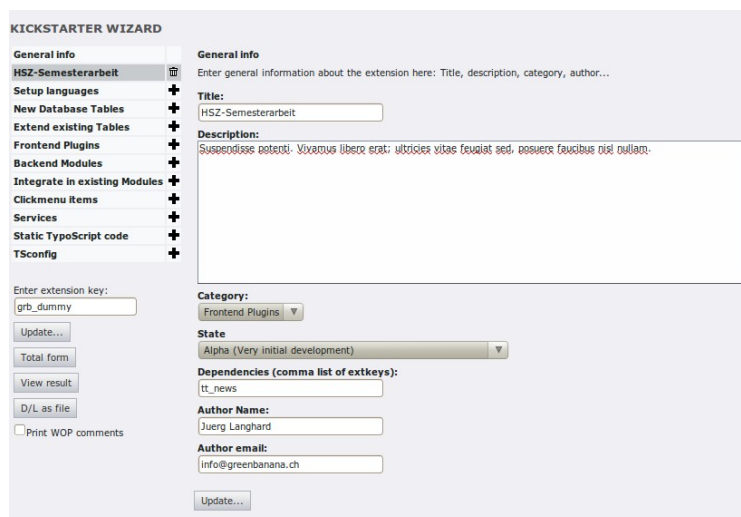
Die herkömmliche Art der Extension-Programmierung beginnt mit der Installation des Kickstarters. Diese TYPO3-Erweiterung vereinfacht die Arbeit des Programmierers zu Beginn der Entwicklung enorm:



Nach der Installation steht im TYPO3-Backend, im Extension Manager, ein Kickstarter zur Verfügung, der hilft, die nötigen Dateien, Ordner und Datenbanktabellen anzulegen. Schrittweise wird folgender Ablauf durchlaufen:

General Info

Die Eingaben unter „General info“ dienen dazu, das File `ext_emconf.php` mit Information zu befüllen. Diese Datei enthält alle wichtigen Eckdaten der Extension. Dazu gehört der Name, eine Beschreibung, Abhängigkeiten/Konflikte zu anderen Extensionen, den Name des Entwicklers, um nur die wichtigsten zu nennen.



Auszug aus `ext_emconf.php`

```
<?php

#####
# Extension Manager/Repository config file for ext "grb_dummy".
#
# Auto generated 29-05-2010 11:58
#
# Manual updates:
# Only the data in the array - everything else is removed by next
# writing. "version" and "dependencies" must not be touched!
#####

$EM_CONF[$_EXTKEY] = array(
    'title' => 'HSZ-Semesterarbeit',
    'description' => 'Suspensio potest. Vivamus libero erat: ultricies vitae feugiat sed, posuere faucibus nisl nullam.',
    'category' => 'plugin',
    'author' => 'Juerg Langhard',
    'author_email' => 'info@greenbanana.ch',
    'dependencies' => 'tt_news',
    'conflicts' => '',
    'author_company' => '',
);

?>
```

Setup languages

Unter „Setup languages“ lässt sich definieren, in welche Sprachen die Extension übersetzt werden soll. Die Wahl optionaler Sprachen wirkt sich dann so auf die weitere Arbeit mit dem Extension-Kickstarter aus, dass alle \$TCA-Felder (für die Anzeige im Backend), die Datenbank-Tabellen und -Felder sowie für die Beschreibungen der Extension Eingabefelder für die Übersetzung angezeigt werden.

Technisch gesehen wird lediglich die Datei `locallang_db.xml` erweitert um weitere `languageKey`'s:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<T3locallang>
  <meta type="array">
    <type>database</type>
    <description>Language labels for database tables/fields 'grb_dummy'</description>
  </meta>
  <data type="array">
    <languageKey index="default" type="array">
      <label index="tt_content.list_type_pi1">HSZ-Semesterarbeit-Plugin</label>
      <label index="tx_grbdummy_table">Player</label>
    </languageKey>
    <languageKey index="de" type="array">
      <label index="tx_grbdummy_table">Spieler</label>
    </languageKey>
  </data>
</T3locallang>
```

New Database Tables und Extend existing Tables

Unter „New Database Tables“ und „Extend existing Tables“ können die nötigen Tabellen und Tabellenfeld mit einem einfachen GUI ohne SQL-Kenntnisse angelegt werden.

Es wird dabei automatisch die Datei `ext_tables.sql` erstellt, die einen Datenbank-Dump der Extension ist.

```
#
# Table: 'tx_grbdummy_table'
#
CREATE TABLE tx_grbdummy_table (
  uid int(11) NOT NULL auto_increment,
  pid int(11) DEFAULT '0' NOT NULL,
  tstamp int(11) DEFAULT '0' NOT NULL,
  crdate int(11) DEFAULT '0' NOT NULL,
  cruser_id int(11) DEFAULT '0' NOT NULL,
  deleted tinyint(4) DEFAULT '0' NOT NULL,
  hidden tinyint(4) DEFAULT '0' NOT NULL,
  images text,

  PRIMARY KEY (uid),
  KEY parent (pid)
);
```

FIELD: images

Field name: (Remove: ☐)

Field title: [English]

[German]

Field type: ☐ Is Exclude-field (What is this?)

Field title

*

Extensions:

Max number of files

Max filesize allowed (kb)

Size of selector box

Field title

☐ Show thumbnails

Frontend- Backend Plugins

Unter „Frontend Plugins“ und/oder „Backend Modules“ kann definiert werden, was die Erweiterung darstellen soll. Es werden dann die nötigen Files angelegt und die Plugins und Module in den entsprechenden Files eingetragen, damit Sie von TYPO3 später richtig geladen werden können.

Frontend-Plugin

In der Datei ext_localconf.php wird folgendes eingetragen:

```
t3lib_extMgm::addPtoST43($_EXTKEY, 'pil/class.tx_grbdummy_pil.php', '_pil', 'list_type', 1);
```

Backend-Modul

In der Datei ext_tables.php wird mit folgendem Code-Ausschnitt zuerst überprüft, ob der Aufruf aus dem Backen kommt, und falls ja wird das Modul „mod1“ geladen:

```
if (TYPO3_MODE == 'BE') {
    t3lib_extMgm::addModulePath('web_txgrbdummyM1', t3lib_extMgm::extPath($_EXTKEY) . 'mod1/');
    t3lib_extMgm::addModule('web', 'txgrbdummyM1', '', t3lib_extMgm::extPath($_EXTKEY) . 'mod1/');
}
```

Static TypoScript code / Tsconfig

Bei den letzten zwei Punkten können die statischen TypoScript-Files angelegt werden. Es handelt sich dabei um Textfiles, die je nach Konfiguration automatisch oder manuell bei der Installation der Erweiterung geladen werden. Sie befinden sich immer unterhalb des Extension-Verzeichnis static/ und heissen setup.txt und constants.txt

Fertigstellen und laden der Erweiterung

Sind alle Anpassungen gemacht, kann die Erweiterung direkt aus dem Kickstarter in die laufende TYPO3-Installation importiert und geladen werden.

ERGEBNIS DES ERWEITERUNGSIMPORTS

Erweiterung importiert

Verzeichnis erstellt: /home/www/p116189/html/typo3/typo3conf/ext/grb_dummy/
ext_emconf.php: /home/www/p116189/html/typo3/typo3conf/ext/grb_dummy/ext_emconf.php
Installationsyp: Lokal

Cache-Dateien wurden entfernt und werden beim nächsten Zugriff neu erstellt.

Tabellen hinzufügen

☒ CREATE TABLE tx_grbdummy_table (
 uid int(11) NOT NULL auto_increment,
 pid int(11) NOT NULL default '0',
 tstamp int(11) NOT NULL default '0',
 crdate int(11) NOT NULL default '0',
 cruser_id int(11) NOT NULL default '0',
 deleted tinyint(4) NOT NULL default '0',
 hidden tinyint(4) NOT NULL default '0',
 images text,
 PRIMARY KEY (uid),
 KEY parent (pid)
);

Hochladeverzeichnis erstellen

Die Erweiterung erfordert das Vorhandensein des Hochladeverzeichnisses 'uploads/tx_grbdummy/'.



Erstellen des Verzeichnisses 'uploads/tx_grbdummy/': ☒

Aktualisierungen durchführen

Erweiterung wurde erfolgreich installiert

Nachdem „Aktualisierung durchführen“ gedrückt wurde sind alle nötigen Datenbank-Tabellen und -Felder erstellt worden und die Erweiterung wurde im File „typo3conf/localconf.php“ eingetragen.

Im Extension-Manager sieht das dann folgendermassen aus:

 **HSZ-Semesterarbeit** **grb_dummy** **0.0.0**  **Lokal** **Alpha**

Anhang II: Ajax im Backend

Der Datenaustausch mittels asynchronem Javascript (AJAX) ist sowohl im Backend wie auch im Frontend möglich. Er unterliegt jedoch genauen Vorgaben, damit es den Coding Guidelines von TYPO3 entspricht.

Grundsätzlich werden alle Ajax-Requestes die aus dem TYPO3-Backend ausgelöst werden, in der TYPO3-Core-Datei `typo3/ajax.php` verarbeitet.

Filesystem

Um die Erweiterung für Backend-Ajax-Aufrufe vorzubereiten, benötigt man zusätzlich folgende Dateien:

- Ajax-Controller (Beispiel: `AjaxController.php`)
- Ajax-Javascript (Beispiel: `backend-main.js`)
- Template (Beispiel: `overview.html`)

Javascript Framework

Im Backend empfiehlt sich die Verwendung des Javascript Frameworks ExtJS, da es auch für die Darstellung des zukünftigen TYPO3-Backend (ab der Version 5.0) zum Einsatz kommt.

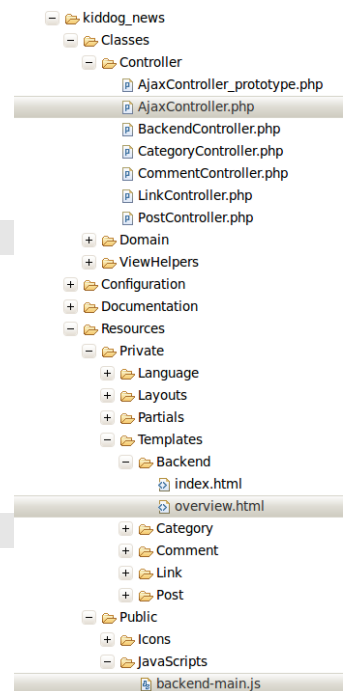
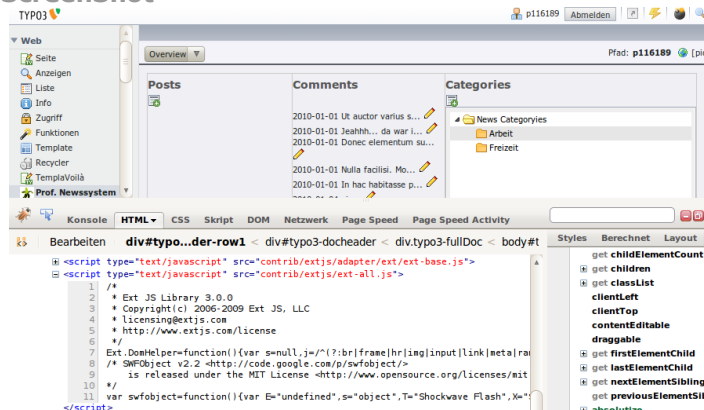
Einbinden von ExtJS mit Fluid

In Fluid lässt sich das Javascript-Framework mit dem Parameter „`loadExtJs=true`“ einbinden.

Beispiel:

```
<f:be.container loadExtJs="true" enableExtJsDebug="true"
  addCssFile="{f:uri.resource(path: 'Styles/backend-main.css')}"
  addJsFile="{
    f:uri.resource(path: 'JavaScripts/backend-main.js')
  }">
  <f:render section="content" />
</f:be.container>
```

Screenshot



Ajax-Methode registrieren

Damit beim Ajax-Request eine bestimmte Methode aufgerufen wird, müssen sogenannte ajaxID's in der Datei ext_localconf.php registriert werden. Diese ID's können per GET-Parameter in der Datei typo3/ajax.php angesprochen werden und steuern so den Aufruf der richtigen Datei mit der entsprechenden Methode oder Funktion.

Beispiel der Datei ext_localconf.php

```
/**
 * Registration of ajax-methodes
 */

// deleteCategoryById
$TYPO3_CONF_VARS['BE']['AJAX']['Tx_KiddogNews_Controller_AjaxController::deleteCategoryById'] =
    'EXT:kiddog_news/Classes/Controller/AjaxController.php:
    Tx_KiddogNews_Controller_AjaxController->deleteCategoryById';

// updateCategoryById
$TYPO3_CONF_VARS['BE']['AJAX']['Tx_KiddogNews_Controller_AjaxController::updateCategoryById'] =
    'EXT:kiddog_news/Classes/Controller/AjaxController.php:
    Tx_KiddogNews_Controller_AjaxController->updateCategoryById';
```

Aufruf-Beispiel

Die Implementation ist somit abgeschlossen und kann zum Beispiel mittels Folgendem Ext.Ajax.request aufgerufen werden:

Auszug aus der Datei backend-main.js

```
function deleteNode() {
    Ext.Ajax.request({
        url : 'ajax.php',
        method: 'GET',
        params : {
            'ajaxID': 'Tx_KiddogNews_Controller_AjaxController::deleteCategoryById'
            , 'tx_kiddognews_ajax[uid]': catUid
        }
        [...]
    });
};
```

Dieser Aufruf löste ein Request mit den GET-Parameter 'ajaxID' und 'tx_kiddognews_ajax[uid]' aus. Dieser wird dann von der Datei typo3/ajax.php ausgewertet und an die Datei "AjaxController.php" und deren Methode „deleteCategoryById“ weiter geleitet, wo die Anfrage abgearbeitet wird.

Auszug aus der Datei AjaxController.php

```
<?php
class Tx_KiddogNews_Controller_AjaxController{

    // Parameters
    protected $tx_kiddognews_ajax;
    public function __construct(){
        $this->tx_kiddognews_ajax = t3lib_div::_GP('tx_kiddognews_ajax');
    }

    /**
     * Delete category by given uid ($_GP[tx_kiddognews_ajax]['uid'])
     *
     */
    public function deleteCategoryById(){
        // Mark category as deleted
        $res = $GLOBALS['TYPO3_DB']->exec_UPDATEquery('tx_kiddognews_domain_model_category',
            'uid='.$this->tx_kiddognews_ajax['uid'], array('deleted' => 1));
        echo $res;
    }
}
```


Anhang III: TypoScript-Parameter

Konfigurationen mittels TypoScript

Mittels TypoScript ist es möglich Variablen und Konfigurationen zu definieren, welche innerhalb TYPO3-Extensions abgerufen werden können. Dies hat den grossen Vorteil, dass Entwickler mit Variablen arbeiten können, welche bei der Implementation der Erweiterung an die Gegebenheiten des jeweiligen Projektes angepasst werden können.

TYPOSCRIPT

```
plugin.tx_kiddognews{
    settings {
        group = 1
    }
}
```

PHP (Controller)

```
$this->settings['group'];
```

Beispiel

Abfrage, ob ein Frontend-User eingeloggt ist, und der Benutzergruppe mit der UID=1 angehört.

```
if($GLOBALS['TSFE']->fe_user->user['usergroup'] == $this->settings['group']){
    $feEditing = TRUE;
}
```

Anhang IV: Mini-Cheat-Sheet Fluid

Wichtige Funktionen

Texte kürzen (f:format.crop)

```
<f:format.crop maxCharacters="100">
    {post.post}
</f:format.crop>
```

Bild rendern (f:image)

```
<f:image src="{user.image}" alt="Benutzerbild von {user.name}" width="120" />
```

Verlinkungen

Action

```
<f:link.action action="show">action
link</f:link.action>
```

E-Mail

```
<f:link.email email="{comment.eMail}" />
```

Files

```
<f:link.external
uri="uploads/tx_kiddognews/{file.file}"
target="_blank">{file.title}</f:link.external>
```

Externe URL

```
<f:link.external uri="{comment.website}"
target="_blank">{comment.website}</f:link.external>
```

Formulare

Select (Array)

Select-Ausgabe eines einfachen Array (\$type), welches vom Controller übergeben wurde.

FLUID:

```
<label for="type">Type</label>
<f:form.select property="type" options="{type}" optionValueField="uid"
optionLabelField="label"><select><option>dummy</option></select></f:form.select><br />
```

FRONTEND:



CONTROLLER:

```
$this->view->assign('type', array( 0 => 'Post', 1 => 'Link'));
```

Fehlermeldungen

RuntimeException

Duplicate variable declarations

Diese Fehlermeldung sagt aus, dass im verwendeten Template zweimal die selbe Variable vor kommt. Dies geschieht relativ schnell, wenn mit Partial gearbeitet wird die for-Schleife haben.

Beispiel:

```
{post.title}
<f:if condition="{post.relatedPosts}">
  <ul>
    <h3>RelatedPosts:</h3>
    <f:for each="{post.relatedPosts}" as="post">
      <li>{post.title}</li>
    </f:for>
  </ul>
</f:if>
```

```
<!-- Ausgabe des Titels -->
```

```
<!-- Gleiche Variable die den Fehler verursacht -->
```

TYPO3 

RuntimeException

Duplicate variable declarations!

Tx_Fluid_Core_Parser_Exception

Cannot cast object of type "DateTime" to string.

Ein Objekt, das im Domain-Model als DateTime deklariert wurde, muss im View folgendermassen ausgegeben werden:

```
<f:format.date>
  {dateObject}
</f:format.date>
```

TYPO3 

Tx_Fluid_Core_Parser_Exception

Cannot cast object of type "DateTime" to string.

Beispiel:

Wird das DateTime-Objekt im View direkt aufgerufen erscheint folgende Fehlermeldung:

Required argument "src" was not supplied.

Wird innerhalb von Fluid ein Bild gerendert, sind zwingend die Angaben „src“ (Destination vom Bild) und „alt“ (Alt-Image-Text) anzugeben. Andernfalls kommt es zu folgender Fehlermeldung:

TYPO3 

Tx_Fluid_Core_Parser_Exception

Required argument "src" was not supplied.

Ein korrektes Beispiel könnte so aussehen:

```
<f:image src="{user.image}" alt="Benutzerbild von {user.name}" width="120" />
```

Quellenverzeichnis

Bücher

Webdatenbank-Applikationen mit PHP und MySQL

Hugh E. Williams & David Lane

O'REILLY

- 2004 – kartoniert - 857 Seiten
- ISBN 3-89721-387-7

TYPO3 4.0: Das Handbuch für Entwickler

Kai Laborenz, Andrea Ertel, Thomas Wendt, Prakash Dussoye, Elmar Hinz

Galileo Press GmbH

- Juni 2006 - gebunden - 808 Seiten
- ISBN: 3898428125
- EAN: 9783898428125

Certified TYPO3 Integrator

Patrick Lobacher

Open Source Press

- April 2009 - kartoniert - 356 Seiten
- ISBN: 3937514783
- EAN: 9783937514789

TYPO3-Extensions

Alexander Ebner, Patrick Lobacher, Bernhard Ulbrich

Hanser Fachbuchverlag

- Mai 2010 - gebunden - XVIII434
- ISBN: 3446415572
- EAN: 9783446415577

PHP objektorientiert

Peter Lavin

Hanser Fachbuchverlag

- November 2006 – gebunden - 224 Seiten
- ISBN-10: 3-446-40762-6
- ISBN-13: 978-3-446-40762-6

PHP Design Patterns

Stephan Schmidt

O'Reilly Vlg. GmbH & Co.

- Februar 2009 - gebunden - XVI489
- ISBN: 389721864X
- EAN: 9783897218642

UML 2 glasklar

Chris Rupp, Stefan Queins, Barbara Zengler

- Hanser Fachbuchverlag
- August 2007 - gebunden - XI554
- ISBN: 3446411186
- EAN: 9783446411180

Entwurfsmuster von Kopf bis Fuss

O'Reilly Vlg. GmbH & Co.

- Dezember 2005 - kartoniert - 672 Seiten
- ISBN: 3897214210
- EAN: 9783897214217

Zeitschriften

t3n .open .web .business

yeebase media GbR
Expo Plaza 3
30539 Hannover

Tel.: +49 (0)511 590 27 99-0

Fax.: +49 (0)511 590 27 99-8

E-Mail: verlag@yeebase.com

URL: <http://www.yeebase.com>

Nr. 18: Kategorien „Content Management“

Titel: Extension-Entwicklung mit Extbase und Fluid

Autor: Patrick Lobacher, freier Entwickler, Autor, Mitglied der TYPO3 Association

Website: www.typofaktum.de

Vortrags- und MicroDokumentationen

Get into the FLOW with Extbase

Jochen Rauch

<http://www.slideshare.net/jocrau/get-into-the-flow-with-extbase-2004846>

MVC for TYPO3 4.3 with extbase

Sebastian Kurfürst

<http://www.slideshare.net/skurfuerst/mvc-for-typo3-43-with-extbase>

Workshop Extension-Entwicklung mit Extbase und Fluid

Sebastian Kurfürst

<http://www.slideshare.net/skurfuerst/workshop-extensionentwicklung-mit-extbase-und-fluid>

Fluid - The Zen of Templating

Sebastian Kurfürst

<http://www.slideshare.net/skurfuerst/fluid-the-zen-of-templating>

Online

TYPO3-Core-API

Copyright © by TYPO3 Core Development Team

Published under the Open Content License available from

<http://www.opencontent.org/opl.shtml>

Dokument-Link:

http://typo3.org/documentation/document-library/core-documentation/doc_core_api/4.2.0/view