Der Einstieg in die Extension-Entwicklung

2. Auflage

Zukunftssichere

TYPO3-Extensions *mit* Extbase & Fluid

Jochen Rau, Sebastian Kurfürst & Martin Helmich

O'REILLY®

2. Auflage

Zukunftssichere TYPO3-Extensions mit Extbase & Fluid

Jochen Rau, Sebastian Kurfürst & Martin Helmich

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
Tel.: 0221/9731600
Fax: 0221/9731608
E-Mail: kommentar@oreilly.de

1. Auflage 2010
2. Auflage 2014

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://d-nb.ddb.de* abrufbar.

Lektorat: Inken Kiupel & Alexandra Follenius, Köln
Korrektorat: Sibylle Feldmann, Düsseldorf
Satz: III-satz, Husby, *www.drei-satz.de*
Umschlaggestaltung: Michael Oreal, Köln
Produktion: Karin Driesen, Köln
Belichtung, Druck und buchbinderische Verarbeitung:
Druckerei Kösel, Krugzell; *www.koeselbuch.de*

ISBN: 978-3-95561-469-0

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Einführung

TYPO3 ist ein mächtiges und ausgereiftes Content-Management-System, das sich durch eine Vielzahl an Funktionen und eine hohe Flexibilität auszeichnet. Die Architektur der aktuell verwendeten Version 6 und die ihr zugrunde liegenden Programmiertechniken entsprechen aber in weiten Teilen dem, was um die Jahrtausendwende Stand der Technik war. Im Jahr 2006 fiel deshalb – nach einer gründlichen Analyse des Status quo – die Entscheidung, TYPO3 von Grund auf neu zu schreiben. Eine Aufteilung in einen Framework-Teil und in das eigentliche Content-Management-System TYPO3 schien sinnvoll. Heute hat sich das Framework TYPO3 Flow bereits am Markt etabliert. Das darauf aufbauende Content-Management-System – ursprünglich geplant als TYPO3 v5 – nimmt heute in Form von TYPO3 Neos als vollkommen eigenständiges Produkt Gestalt an. In diese Phase der Neuorientierung fiel auch die Geburtsstunde von Extbase und Fluid.

Extbase ist ein PHP-basiertes Framework, das Entwickler darin unterstützt, saubere und gut wartbare TYPO3-Extensions zu schreiben, die darüber hinaus zukunftssicher sind, weil eine Portierung nach TYPO3 Flow erleichtert wird. Die Template-Engine Fluid sorgt dafür, dass die Oberfläche der Extension leicht individuell gestaltet werden kann.

Extbase gibt eine klare Trennung verschiedener Zuständigkeiten vor, die eine einfache Wartung des Codes möglich macht, weil Letzterer in Modulen gekapselt wird. Durch diesen modularen Aufbau sinken die Entwicklungszeit für Erst- und Anpassungsentwicklungen und die damit mittelbar und unmittelbar verbundenen Kosten. Extbase entlastet Entwickler außerdem bei sicherheitskritischen und wiederkehrenden Aufgaben, wie beispielsweise der Validierung von Argumenten, der Persistenz der Daten und dem Auslesen der Einstellungen aus TypoScript und FlexForms. Dadurch können sich Entwickler darauf konzentrieren, die Problemstellung ihrer Auftraggeber zu lösen.

Durch die moderne Extension-Architektur und die Verwendung von aktuellen Software-Entwicklungsparadigmen setzt die Nutzung von Extbase allerdings anderes Fachwissen als bisher voraus. Anstatt eine Extension einfach »zusammenzuhacken«, müssen Programmierer nun einige Konzepte wie Domain-Driven Design verstehen und die Exten-

sion vor der Umsetzung genauer planen und modellieren. Im Gegenzug wird durch die Anwendung dieser Konzepte der Quelltext der Extension besser lesbar, flexibler und erweiterbar. Außerdem sind die hier eingesetzten Konzepte auch auf andere Programmiersprachen und Frameworks übertragbar, da die verwendeten Entwicklungsparadigmen wie Domain-Driven Design oder eine Model-View-Controller-Architektur universell einsetzbar sind.

Extensions, die auf Extbase aufbauen, können mit überschaubarem Aufwand zu TYPO3 Neos bzw. TYPO3 Flow portiert werden, da sich die Struktur der Extension, die Namenskonventionen und die verwendeten Schnittstellendefinitionen (APIs) weitgehend gleichen. Somit wird durch Extbase der Übergang hin zu TYPO3 Neos und Flow vereinfacht – wenn Sie Extbase nutzen, werden Sie später leicht auf TYPO3 Flow und Neos wechseln können.

TYPO3 CMS und TYPO3 Neos

In der TYPO3-Community werden zurzeit mehrere Produkte parallel entwickelt, die hier kurz vorgestellt werden sollen. TYPO3 CMS ist der seit Jahren bewährte Zweig von TYPO3, mit dem Hunderttausende Webseiten weltweit verwaltet werden. Zum Erscheinungszeitpunkt dieses Buchs war die Version 6.2 aktuell, und Version 6.3 befand sich in der Entwicklung.

Da die internen Strukturen von TYPO3 CMS organisch gewachsen sind und damit recht unübersichtlich sein können, entschloss sich das TYPO3-Team, parallel zur Pflege der damaligen Version 4 von TYPO3 CMS einen neuen Entwicklungszweig zu beginnen und TYPO3 v5 von Grund auf zu entwickeln. Dabei wurde besonders auf sauberen Code und eine klare und mächtige Infrastruktur geachtet. Es kristallisierte sich schnell heraus, dass zuerst ein Webanwendungs-Framework geschrieben werden sollte, bevor das CMS selbst entwickelt wird. Dieses Webanwendungs-Framework trug den Namen FLOW3 (mittlerweile TYPO3 Flow), und diente als Grundlage von TYPO v5.

Später wurde beschlossen, TYPO3 v5 nicht als Nachfolger von TYPO3 v4, sondern als komplett eigenständiges Produkt weiterzuentwickeln. Um Verwirrungen um die Versionsnummern zu vermeiden, wurde für TYPO3 CMS daher die Versionsnummer 5 übersprungen, und das ehemalige TYPO3 v5 existiert nun unter dem Namen TYPO3 Neos.

Wir hoffen, dass Extbase und Fluid für Sie der Einstieg in eine neue Welt des Programmierens sind und Sie sich mit neuen Konzepten von TYPO3 Flow schnell vertraut machen können. Sie werden anfangs viele neue Konzepte lernen, aber mit der Zeit feststellen, dass Sie mit Extbase und Fluid viel produktiver arbeiten können als bisher. Mit etwas Glück geraten Sie vielleicht auch in den »Flow«, den wir beim Entwickeln von und mit Extbase und Fluid gespürt haben.

Nicht nur Sie als Entwickler können von Extbase profitieren: Auch das TYPO3 Flow-Entwicklerteam kann mit Extbase viele Konzepte auf ihre Praxistauglichkeit hin überprüfen und gleichzeitig Fehler und Inkonsistenzen beseitigen, die in Extbase und Fluid auftreten.

Eine kurze Geschichte von Extbase und Fluid

Nachdem der Startschuss für die Neuentwicklung von TYPO3 v5 und FLOW3 als dem darunterliegenden Framework gefallen war, verlief die Entwicklung von TYPO3 v4 und TYPO3 v5 zunächst weitgehend unabhängig voneinander. Im Oktober 2008 trafen sich die Kernentwickler der beiden Entwicklungslinien zu den *Transition Days* in Berlin. Dort sollte eine gemeinsame Vision und Strategie für den Übergang von der heutigen TYPO3 Version 4 zur kommenden Version 5 erarbeitet werden. Die Kernpunkte dieser Strategie wurden zu einem Manifest zusammengefasst (siehe den Kasten »The Berlin Manifesto«).

The Berlin Manifesto

We, the participants of the TYPO3 Transition Days 2008 state that ...

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

Signed by:

Patrick Broens, Karsten Dambekalns, Dmitry Dulepov, Andreas Förthner, Oliver Hader, Martin Herr, Christian Jul Jensen, Thorsten Kahler, Steffen Kamper, Christian Kuhn, Sebastian Kurfürst, Martin Kutschker, Robert Lemke, Tobias Liebig, Benjamin Mack, Peter Niederlag, Jochen Rau, Ingo Renner, Ingmar Schlecht, Jeff Segars, Michael Stucki, Bastian Waidelich

Vor dem Hintergrund dieses Manifests fiel die Entscheidung, zwei Teile von TYPO3 v4 neu zu implementieren:

- Einen modernen Nachfolger für die Basisklasse `tslib_piBase`, auf der heute die Mehrzahl der über 3.600 Extensions für TYPO3 aufbaut. Daraus ist Extbase entstanden.
- Eine neue Template-Engine zur Ausgabe von Daten, die Flexibilität, Einfachheit und einfache Erweiterbarkeit miteinander vereint: Fluid.

Schon auf den Transition Days hatten Bastian Waidelich und Sebastian Kurfürst diskutiert, wie eine neue Template-Engine aussehen und funktionieren könnte, und kurz darauf haben sie einen Prototyp implementiert. Die Entwicklung von Extbase begann zwei Monate später im Rahmen eines Treffens einiger Core-Mitglieder in Karlsruhe. Dort einigte man sich auch darauf, sich so nahe wie möglich an den Ideen, der Architektur und den Schnittstellen von FLOW3 zu orientieren.

Darauf folgte eine intensive Entwicklungszeit, in der Jochen Rau den größten Teil von Extbase entwickelt hat, während Sebastian Kurfürst immer wieder Code-Reviews durchgeführt hat. Außerdem hat Fluid in dieser Zeit das Betastadium erreicht.

Die erste öffentliche Präsentation von Extbase fand im März 2009 auf der *T3BOARD09* in Laax (CH) statt. Bei stürmischem Wetter, 2.228 Meter über dem Meeresspiegel, konnten Core-Entwickler und Interessierte den aktuellen Stand von Extbase begutachten. In einer angeregten Diskussion wurden die letzten offenen Punkte, wie Namenskonventionen oder der Name des Frameworks, geklärt. Außerdem fiel auch die Entscheidung, Fluid doch schon in TYPO3 4.3 zu integrieren, anstatt wie vorher geplant erst in Version TYPO3 4.4.

Daraufhin entwickelte das Fluid-Team in wenigen Tagen ein kleines Programm (namens *Backporter*), das den Code von Fluid für FLOW3 in Code für TYPO3 v4 umwandelt. Somit gab es am Ende der *T3BOARD09* die erste lauffähige Version von Fluid auf TYPO3 v4, und es war nun recht einfach, Fluid für FLOW3 synchron mit der Version für TYPO3 v4 zu halten.

Das erste Mal präsentiert wurden Extbase und Fluid im April 2009 auf der amerikanischen *TYPO3 Conference* in Dallas, und einen Monat später auf den *TYPO3 Developer Days* in Elmshorn bei Hamburg. Daraufhin gab es viele positive Rückmeldungen und konstruktive Kritik aus der Community.

Während der nächsten Monate floss viel Zeit in Details: Die Syntax von Fluid wurde verbessert, ViewHelper wurden geschrieben, und die Persistenzschicht von Extbase wurde mehrfach verbessert und aufgeräumt. Es wurden Funktionalitäten für die Sicherheit des Frameworks implementiert, das MVC-Framework wurde aufgeräumt, und weitere Funktionen wurden implementiert, die im Praxiseinsatz benötigt wurden.

Zur *T3CON09* im Herbst 2009 konnten Extbase und Fluid in einer Betaversion präsentiert werden. Danach wurden nur noch Fehler korrigiert, jedoch keine wesentlichen neuen Funktionalitäten implementiert. Mit dem Release von TYPO3 4.3.0 im November 2009 sind Extbase und Fluid schließlich in den TYPO3 Core aufgenommen worden und sind damit auf jeder TYPO3-Installation verfügbar.

Nach einer wohlverdienten Pause des Entwicklerteams nach dem Release begann die Arbeit wieder mit kleineren Bugfixes und einer Roadmap für weitere Funktionalitäten. So wurden viele Details im Framework verbessert, und außerdem wurde die Persistenzschicht erneut verschlankt und aufgeräumt.

Auf der *T3BOARD12* im März 2012 beschloss das TYPO3-Entwicklerteam schließlich, die zu dem Zeitpunkt bereits seit vier Jahren in Entwicklung befindliche TYPO3-Version 5 in ein eigenständiges Produkt mit dem Arbeitsnamen TYPO3 Phoenix auszulagern. Um Verwirrung bezüglich der Versionsnummer zu vermeiden, wurde die Versionsnummer 5 anschließend übersprungen, und im November 2012 erschien die TYPO3-Version 6.0 als direkter Nachfolger von TYPO3-Version 4.7.[1]

Im Herbst 2012 wurde im Zuge einer Umstrukturierung der TYPO3-Markenlandschaft beschlossen, TYPO3 unter dem Namen TYPO3 CMS, FLOW3 unter dem Namen TYPO3 Flow und TYPO3 Phoenix unter TYPO3 Neos weiterzuführen.

An wen sich dieses Buch richtet

Dieses Buch richtet sich an TYPO3-Extension-Entwickler, die Grundkenntnisse in der PHP-Programmierung und im Umgang mit der Administration von TYPO3 mitbringen. Es bietet Ihnen einen kompakten Einstieg in das Framework Extbase und die Template-Engine Fluid. Wir wenden uns speziell an:

- Einsteiger, die von Anfang an Extbase und Fluid als Grundlage für eigene Extensions verwenden möchten
- erfahrene Entwickler, die sich vor einem neuen Projekt in Extbase und Fluid einarbeiten möchten
- Entwickler, die ihre bestehenden Extensions frühzeitig auf TYPO3 Flow portieren möchten
- Entscheider, die einen technischen Überblick über das neue Framework erhalten möchten

Aufbau dieses Buchs

Das Buch ist in zehn Kapitel und drei Anhänge unterteilt. Die Kapitel behandeln im Einzelnen folgende Themen:

Kapitel 1, *Installation*, führt Sie durch die Installation von Extbase und Fluid. Um die Extension-Entwicklung möglichst effektiv zu gestalten, geben wir hier einige Hinweise zu Entwicklungsumgebungen sowie Tipps und Tricks zum Debugging.

Kapitel 2, *Grundlagen*, beginnt mit einem Überblick über die Konzepte der objektorientierten Programmierung, da diese für die Arbeit mit Extbase essenziell sind. Danach beschäftigen wir uns mit Domain-Driven Design, einem Programmierparadigma, das

1 Siehe hierzu auch *http://typo3.org/news/article/the-typo3-core-team-is-giving-a-short-summary-about-the-upcoming-releases-of-typo3/*

durch Extbase optimal unterstützt wird. Im Anschluss daran lernen Sie das Entwurfsmuster Model-View-Controller kennen, das die technische Grundlage einer jeden Extbase-Extension bildet. Abgerundet wird das Kapitel durch eine Einführung ins Test-Driven Development.

Kapitel 3, *Reise durch das Blog-Beispiel*, soll Ihnen ein Gespür dafür geben, wie die in Kapitel 2 dargestellten Konzepte in Extbase umgesetzt wurden. Anhand einer bestehenden Beispiel-Extension erklären wir, wie ein Blog-Beitrag angelegt wird und die verschiedenen Stationen des Systems durchläuft, bis er ausgegeben wird.

In Kapitel 4, *Eine erste Extension anlegen*, stellen wir Ihnen eine minimale Extension vor, mit der Daten, die im TYPO3-Backend verwaltet werden, im Frontend ausgegeben werden.

Kapitel 5, *Die Domäne modellieren*, zeigt anhand eines Praxisbeispiels, wie mit Domain-Driven Design ein Modell geplant und umgesetzt werden kann.

Wenn das Domänenmodell steht, muss die dazu nötige TYPO3-Infrastruktur, also die Datenbanktabellen und Backend-Bearbeitungsformulare, angelegt werden. Alle hierzu nötigen Informationen finden Sie in Kapitel 6, *Die Persistenzschicht einrichten*.

Nachdem Kapitel 5 und 6 die Modell-Schicht näher beschrieben haben, geht es in Kapitel 7, *Den Ablauf mit Controllern steuern,* um die Abläufe innerhalb der Extension. Diese werden in der Controller-Schicht implementiert.

Nun fehlt nur noch die Ausgabe-Schicht der Extension, der sogenannte *View*. In Kapitel 8, *Die Ausgabe mit Fluid gestalten,* wird Ihnen Fluid nähergebracht und werden seine Funktionen anhand verschiedener Beispiele erläutert. Abschließend werden die vorher erklärten Funktionen kombiniert und am Beispielprojekt demonstriert.

Kapitel 9, *Mehrsprachigkeit, Validierung und Sicherheit*, widmet sich fortgeschritteneren Themen und Aufgaben. Dazu zählen die mehrsprachige Umsetzung einer Extension, die Validierung von Daten und die Berücksichtigung von Sicherheitsaspekten.

Kapitel 10, *Ausblick*, greift spannende, noch in der Entwicklung befindliche Funktionalitäten auf. Der Fokus liegt hier insbesondere darauf, wie Sie eigene Extensions erweiterbar halten, und der Verwendung von Extbase im TYPO3-Backend. Außerdem gibt dieses Kapitel einen Ausblick auf TYPO3 Flow und geht auf die Portierung von TYPO3-Extensions in TYPO3 Neos ein.

Extbase nutzt weitestgehend die Konventionen von TYPO3 Flow. Diese finden Sie in Anhang A, *Coding Guidelines,* übersichtlich zusammengefasst.

Anhang B, *Referenz für Extbase*, beinhaltet eine Übersicht über wichtige Extbase-Konzepte und eine alphabetische Übersicht der APIs .

In Anhang C, *Referenz für Fluid*, finden Sie eine Referenz aller mit Fluid mitgelieferten ViewHelper und die APIs, die Sie zum Schreiben eigener ViewHelper benötigen.

Code-Beispiele

Auf der Website des Verlags unter *http://examples.oreilly.de/german_examples/typo3ext2ger/* finden Sie den im Buch beschriebenen Code sowie die verwendeten Extensions zum Download.

Typografische Konventionen

Dieses Buch verwendet die folgenden typografischen Konventionen:

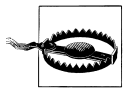
Kursiv
: Wird für wichtige Begriffe, Programm- und Dateinamen, URLs, Ordner und Verzeichnispfade, Menüs, Optionen und zur Hervorhebung verwendet.

`Nichtproportionalschrift`
: Wird für Befehle, Optionen, Variablen, Attribute, Klassen, Namensräume, Methoden, Module, Eigenschaften, Parameter, Werte, Objekte, Ereignisse, Event-Handler, ViewHelper, XML- und HTML-Elemente und die Ausgabe von Befehlen verwendet.

`Nichtproportionalschrift fett`
: Wird zur Hervorhebung im Code verwendet.

`Nichtproportionalschrift kursiv`
: Wird innerhalb des Codes verwendet, wenn der Benutzer Teile des Codes durch eigene Eingaben oder Werte ersetzen soll.

Dieses Symbol steht für einen Tipp, einen Vorschlag oder einen allgemeinen Hinweis.



Mit diesem Symbol wird auf Besonderheiten hingewiesen, die zu Problemen führen oder ein Risiko darstellen können.

Danksagungen

Allen voran wollen wir unseren Familien und Partnern für das Verständnis und die Unterstützung danken, wenn wir die Nachmittage und Abende anstatt mit ihnen mit dem Buch verbracht haben. Auch unsere Kunden haben das eine oder andere Mal Geduld aufbringen müssen, als wir Extbase entwickelt oder das Buch geschrieben haben, anstatt das Kundenprojekt weiterzuführen.

TYPO3 würde nicht existieren ohne den Einsatz und die Vision von Kasper Skårhøj und ohne die unermüdliche Arbeit aller Beitragenden und besonders des Core Teams. Hier

wollen wir vor allem denjenigen Mitgliedern danken, die für TYPO3 Flow und Neos viele zukunftsweisende Konzepte entdeckt und umgesetzt haben. Extbase wäre ohne diese Inspiration und Vorlage nicht möglich gewesen.

Aber auch bei der Arbeit am Buch konnten wir auf tatkräftige Unterstützung bauen: Wir wollen Patrick Lobacher danken. Er hat den Abschnitt über die objektorientierte Programmierung beigesteuert.

Unserer besonderer Dank gilt unseren Lektorinnen, Alexandra Follenius und Inken Kiupel, die uns unermüdlich Feedback und Kommentare zu unseren Texten gegeben haben und so großen Einfluss auf die Entstehung des Buchs hatten. Außerdem danken wir den vielen unbekannten Helfern bei O'Reilly, die dieses Buch letztendlich umsetzen.

Nicht zuletzt danken wir Ihnen – dafür, dass Sie Extbase und Fluid verwenden wollen!

KAPITEL 1

Installation

In diesem Kapitel wollen wir Ihnen einige Hilfestellungen zur effizienten Einrichtung Ihrer Arbeitsumgebung geben. Zu Beginn gibt es einige Hinweise zum Einrichten des Entwicklungsservers. Danach wird erklärt, wie Sie die beiden TYPO3-Erweiterungen *extbase* und *fluid* installieren können, um die sich das restliche Buch dreht. Außerdem geben wir Ihnen einige Empfehlungen zum Einrichten einer Entwicklungsumgebung (IDE), sodass Sie Codevervollständigung und die in der IDE integrierten Debugger nutzen können. Das Kapitel wird abgerundet durch eine Liste von weiteren hilfreichen TYPO3-Erweiterungen und deren Bezugsquellen.

Wir gehen davon aus, dass Sie Kenntnisse in der Installation von TYPO3 besitzen, und werden uns daher auf die Installation von Extbase und Fluid sowie auf damit im Zusammenhang stehende Komponenten konzentrieren.

Den Server einrichten

Da TYPO3 in der Skriptsprache PHP geschrieben ist, benötigen Sie zur TYPO3-Entwicklung einen Webserver wie Apache mit PHP-Unterstützung (Version 5.3 oder 5.4). Außerdem benötigt TYPO3 eine MySQL-Datenbank zur Datenspeicherung. Falls Sie noch keinen lokalen Entwicklungsserver haben, können wir Ihnen das XAMPP-Paket (*http://www.apachefriends.org/xampp.html*) empfehlen. Dieses installiert Apache, PHP, MySQL sowie einige weitere hilfreiche Tools auf allen gängigen Plattformen (Linux, Windows, Mac OS X). Nun können Sie TYPO3 auf Ihrem Testsystem installieren.

Für Produktivsysteme ist es empfehlenswert, einen PHP Opcode Cache wie *eAccelerator* (*http://eaccelerator.net*) einzusetzen, da dieser den kompilierten PHP-Code zwischenspeichert, wodurch sich die Ladezeit der Skripte oft mehr als halbiert. Standardmäßig speichert eAccelerator die PHP-Kommentare einer Datei nicht. Da Extbase über diese Kommentare jedoch wichtige Informationen erhält, dürfen sie von eAccelerator nicht entfernt werden.

Hierfür müssen Sie eAccelerator mit der option `--with-eaccelerator-doc-comment-inclusion` konfigurieren. Eine vollständige Installation von eAccelerator sieht dann also folgendermaßen aus: Zunächst müssen Sie den Quellcode von eAccelerator herunterladen und auf der Konsole in das Quelltextverzeichnis wechseln. Sie müssen den eAccelerator-Quelltext durch das Kommando `phpize` auf die installierte PHP-Version abstimmen. Dann können Sie die Kompilierung der Quelldateien vorbereiten, indem Sie eAccelerator mit dem Befehl `./configure --with-eaccelerator-doc-comment-inclusion` mitteilen, dass die Quelltextkommentare nicht entfernt werden sollen. Kompilieren Sie jetzt den eAccelerator mithilfe des Befehls `make`. Zu guter Letzt muss eAccelerator nur noch mit `make install` installiert werden. Dieser Schritt muss als Benutzer *root* erfolgen.

Eventuell muss noch die PHP-Konfiguration angepasst werden, sodass eAccelerator geladen wird. Sie können nun in einer TYPO3-Instanz überprüfen, ob die Quelltextkommentare erhalten bleiben. Wechseln Sie hierzu im TYPO3-Backend in das Install Tool. Dort wählen Sie das Untermodul *System environment* aus. Hier sehen Sie unter dem Punkt *Document comment reflection*, ob die PHP-Kommentare erhalten bleiben oder nicht (siehe Abbildung 1-1).

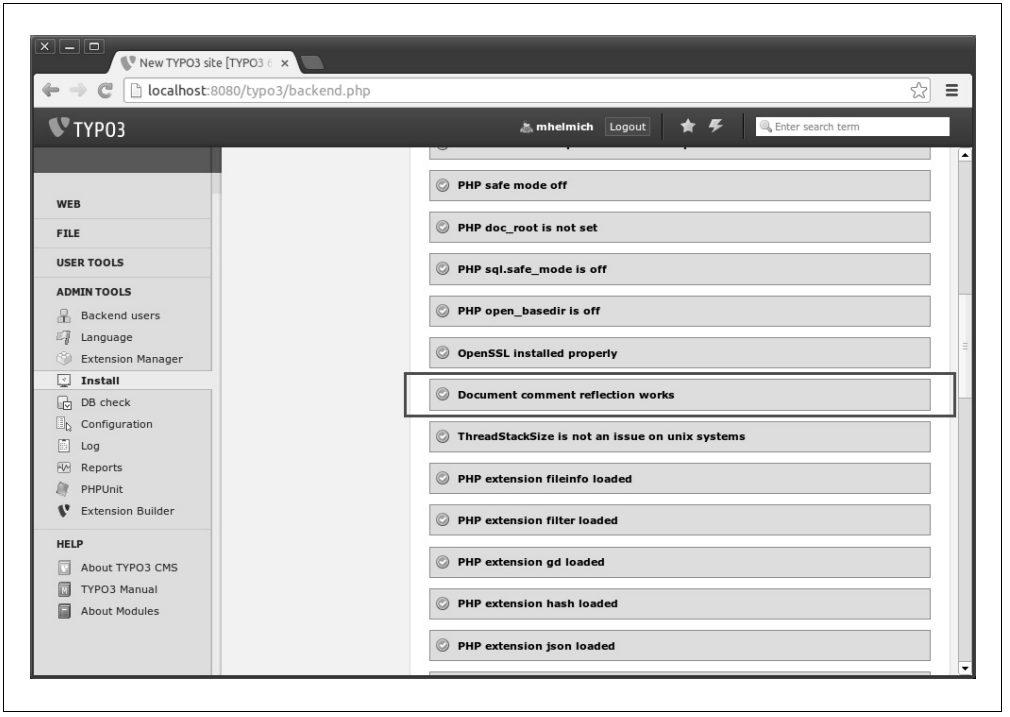


Abbildung 1-1: Im Install-Tool können Sie überprüfen, ob Extbase Zugriff auf die Quelltextkommentare der PHP-Dateien hat.

Extbase und Fluid installieren

Extbase und Fluid werden seit der TYPO3-Version 4.3 mit dem Kern von TYPO3 als sogenannte *System Extensions* ausgeliefert. Seit der TYPO3-Version 4.5 sind Extbase und Fluid zudem integrale Bestandteile des TYPO3-Kerns (sie sind als solche bereits vorinstalliert und können auch nicht deinstalliert werden). Im Unterschied zu früheren Versionen von TYPO3 und Extbase können Sie also sofort mit der Extension-Entwicklung beginnen.

Um zu überprüfen, ob Extbase und Fluid funktionstüchtig sind, werden wir jetzt das Blog-Beispiel, eine Extension, die zum Testen von Extbase und Fluid geschrieben wurde, installieren. Diese Extension stellen wir Ihnen in Kapitel 3 genauer vor.

Die Beispiele in diesem Buch beziehen sich auf Version 1.4 des Blog-Beispiels. Diese war zum Zeitpunkt des Verfassens dieses Buchs noch nicht im Extension Repository verfügbar; Sie finden eine Vorabversion jedoch bei den Downloads zu diesem Buch unter *http://examples.oreilly.de/german_examples/typo3ext2ger/*.

Wählen Sie hierzu im Extension Manager die Schaltfläche *Upload extension .t3x/.zip* und wählen Sie die entsprechende ZIP-Datei aus den Codebeispielen aus. Nach einem Klick auf *Upload* wird die Extension in Ihre TYPO3-Instanz hochgeladen und auch gleich installiert.

Nachdem Sie dies abgeschlossen haben, finden Sie im *Web*-Modul das Untermodul *Blogs*. Falls es nicht im Menü zu sehen ist, sollten Sie das Backend neu laden. Um das Modul nutzen zu können, müssen Sie zunächst im TypoScript-Setup Ihrer *root*-Seite das statische Template *BloxExample setup* inkludieren. Anschließend können Sie im Blog-Modul durch einen Klick auf *Create example data* Beispieldatensätze erzeugen. Das Blog-Beispiel ist nun funktionstüchtig, und Sie können weitere Schritte mit Extbase und Fluid gehen.

Die Entwicklungsumgebung einrichten

Da eine auf Extbase basierende Extension aus vielen Dateien besteht, ist es hilfreich, eine PHP-Entwicklungsumgebung (IDE) anstatt eines einfachen Editors zu verwenden. Eine IDE bietet Ihnen neben Syntax-Highlighting vor allem Codevervollständigung und eine direkte Ansicht der Codedokumentation. Außerdem haben manche Entwicklungsumgebungen einen Debugger integriert, der die Fehlersuche erheblich vereinfachen und beschleunigen kann. Wir werden hier exemplarisch zeigen, wie Sie NetBeans und Eclipse zur Extension-Entwicklung einrichten. Beide Entwicklungsumgebungen haben eine vergleichbare Funktionalität, daher hängt die Entscheidung für die eine oder andere IDE sehr von Ihren persönlichen Vorlieben ab.

Bezugsquellen

Die IDE NetBeans steht für Windows, Linux und Mac OS X zur Verfügung. Wir verwenden NetBeans PHP, die Sie unter *http://netbeans.org* herunterladen können.

Auch Eclipse ist für alle wichtigen Betriebssysteme verfügbar. Die PHP-Entwicklungsumgebung heißt *Eclipse PDT*, und Sie können diese auf *http://eclipse.org/pdt* für alle wichtigen Plattformen herunterladen.

Generell ist es hilfreich, für Extbase und Fluid Projekte in NetBeans bzw. Eclipse anzulegen. So können Sie sich den Quelltext von Extbase bzw. Fluid anschauen und die Codedokumentation lesen, wenn dies nötig ist.

Für NetBeans öffnen Sie dazu im Menü *File* den Eintrag *New Project* und wählen dort unter der Kategorie *PHP* den Eintrag *PHP Application with Existing Sources*. Auf der nächsten Seite des Wizards können Sie das Extension-Verzeichnis von Fluid bzw. Extbase auswählen. Sie finden diese in */pfad-zur-typo3-installation/typo3/sysext/extbase/* bzw. *.../fluid/*.

NetBeans verwendet zur Einrückung in Quelltextdateien standardmäßig Leerzeichen. Da die TYPO3 Coding Guidelines die Verwendung von Tabs definieren, sollten Sie NetBeans dementsprechend umstellen. Öffnen Sie dazu die NetBeans-Einstellungen, und wählen Sie den Unterpunkt *Editor*. Nun sollten Sie unter *Formatting* die Einstellung *Expand Tabs to Spaces* deaktivieren und die Werte für *Number of Spaces per Indent* und *Tab Size* gleich einstellen (beispielsweise auf 4).

In Eclipse können Sie Projekte für Extbase und Fluid folgendermaßen anlegen: Klicken Sie auf *File → New Project*, und wählen Sie *Create project from existing source* aus. Wählen Sie dann den entsprechenden Ordner für Extbase bzw. Fluid aus, und geben Sie dem Projekt einen Namen. Mit dem Klick auf *Finish* wird das Projekt in Eclipse angelegt.

Wenn Sie eine eigene Erweiterung schreiben, sollten Sie auch für diese ein Projekt in Eclipse bzw. NetBeans anlegen.

Autovervollständigung für Extbase und Fluid einrichten

Beim Schreiben von eigenen Extensions werden Sie oft mit den Klassen von Extbase und Fluid arbeiten. Um Tippfehler zu vermeiden, ist daher die Einrichtung der Autovervollständigung hilfreich. Damit werden Ihnen beim Druck der Tastenkombination *Strg + Leerzeichen* Vorschläge für den vollständigen Klassennamen gemacht (siehe Abbildung 1-2). Um diese Funktionalität zu aktivieren, muss angegeben werden, dass das Projekt, das Sie schreiben, von Extbase und Fluid abhängig ist – dann wird die IDE auch Autovervollständigung für Klassen aus Extbase bzw. Fluid ermöglichen.

Abbildung 1-2: Die Autovervollständigung zeigt Ihnen mögliche Klassennamen und die Codedokumentation derselben.

In NetBeans müssen Sie dazu mit der rechten Maustaste auf das Projekt Ihrer Extension klicken. Im aufklappenden Kontextmenü wählen Sie nun *Properties*, um die Projekteigenschaften zu bearbeiten. Wählen Sie nun die Kategorie *PHP Include Path*, und fügen Sie mittels *Add Folder...* die Verzeichnisse von Extbase bzw. Fluid hinzu.

In Eclipse geht dies fast genauso. Klicken Sie auch hier mit der rechten Maustaste auf das Projekt, zu dem Sie die Autovervollständigung hinzufügen wollen, und wählen Sie *Properties*. Nun wählen Sie die Kategorie *PHP Include Path*. Klicken Sie dann auf *Projects*, da Sie eine Referenz auf ein anderes Eclipse-Projekt hinzufügen wollen. Wählen Sie nach dem Klick auf den Button *Add...* die Projekte von Extbase und Fluid aus, die Sie im vorherigen Abschnitt erzeugt haben.

Debugging mit Xdebug

Während das Verwenden von Debuggern in anderen Programmiersprachen fest zum Arbeitsablauf der Programmierung gehört, ist dies in PHP eher eine Randerscheinung: Die meisten PHP-Entwickler betreiben Fehlersuche, indem sie Ausgaben, z.B. mittels `echo` oder `var_dump`, in den Quelltext einbauen und so den Programmablauf nachvollziehen. Gerade bei komplexen Fehlerbildern kann es jedoch hilfreich sein, mit einem echten Debugger das Problem nachzuvollziehen – hier kann man schrittweise durch den Quelltext gehen und beispielsweise direkt den Inhalt von Variablen sehen.

Um ein Debugging zu ermöglichen, benötigen Sie die PHP-Erweiterung *Xdebug* (*http://xdebug.org*). Sie verändert PHP dahin gehend, dass die Ausführung von PHP-Befehlen schrittweise erfolgen kann und die Inhalte von Variablen inspiziert werden können. Die Entwicklungsumgebungen NetBeans und Eclipse verbinden sich mit Xdebug, um eine grafische Darstellung des Debuggings zu ermöglichen.

Doch erst einmal zurück zu Xdebug: Diese PHP-Erweiterung kann auf verschiedene Weisen installiert werden – wir empfehlen entweder die Installation über ein Paketmanagementsystem (wie beispielsweise APT für Debian/Ubuntu oder MacPorts für Mac OS X) oder die Installation aus PECL (PHP Extension Community Library). Letztere kann durch nachfolgenden Konsolenbefehl (als Benutzer *root*) gestartet werden:

```
pecl install xdebug
```

Nach erfolgreicher Installation muss Xdebug noch PHP bekannt gemacht werden. Dazu müssen Sie die Datei *php.ini* anpassen und folgende Zeile ergänzen:

```
zend_extension="/usr/local/php/modules/xdebug.so"
```

Dabei müssen Sie den korrekten Pfad zur Datei *xdebug.so* angeben.

Für Windows können Sie eine fertig kompilierte Xdebug-Version von der *Xdebug*-Website herunterladen.

Nach einem Neustart von Apache sollten Sie nun in der Ausgabe von `phpinfo()` einen Abschnitt *Xdebug* finden. Nun muss XDebug noch konfiguriert werden, sodass es mit Eclipse bzw. NetBeans zusammenarbeitet. Damit sich ein Debugger wie der von NetBeans bzw. Eclipse mit Xdebug verbindet, müssen Sie folgende Einstellungen in der *php.ini*-Konfigurationsdatei setzen:

```
xdebug.remote_enable=on
xdebug.remote_handler=dbgp
xdebug.remote_host=localhost
xdebug.remote_port=9000
```

Nach einem Apache-Neustart sollten Sie die obigen Werte auch in der Ausgabe von `phpinfo()` sehen. Nun müssen Sie die Entwicklungsumgebung noch konfigurieren, damit diese Xdebug korrekt startet. Für NetBeans müssen Sie die Projekteigenschaften öffnen, indem Sie mit der rechten Maustaste auf das zu debuggende Projekt klicken und *Properties* auswählen. Nun müssen Sie die *Run Configuration* ändern: Geben Sie als *Project URL* die Basis-URL an, unter der das TYPO3-Frontend Ihres Testsystems erreichbar ist, also beispielsweise: *http://localhost/typo3v4/*. Klicken Sie dann auf *Advanced...*, und stellen Sie die Einstellung *Debug URL* auf *Ask Every Time*. NetBeans ist jetzt für das Debugging vorbereitet.

Sie können im Quelltext des zu debuggenden Projekts sogenannte *Breakpoints* setzen. An diesen Stellen wird das Programm unterbrochen, und Sie können den Programmzustand ansehen, also beispielsweise den Inhalt von Variablen untersuchen oder das Programm

Zeile für Zeile ablaufen lassen. Um einen Breakpoint zu setzen, klicken Sie auf die Zeilennummerierung in der entsprechenden Quelltextdatei. Die Zeile wird dann rot hervorgehoben.

Zum Starten des Debuggings wählen Sie im Menü den Eintrag *Debug → Debug Project*. Es öffnet sich ein Pop-up-Fenster, in dem Sie die aufzurufende URL eintragen müssen. Empfehlenswert ist, diese direkt aus der Adresszeile Ihres Browsers zu kopieren: Wenn Sie beispielsweise das Blog-Beispiel debuggen wollen, öffnen Sie das Blog-Modul im TYPO3-Backend, klicken mit der rechten Maustaste in den Inhaltsframe und wählen *Aktueller Frame → Frame* in neuem Tab öffnen. Kopieren Sie die URL dieses Frames, und tragen Sie sie bei NetBeans ein. NetBeans öffnet dann den Browser mit der gewünschten URL, und Sie sehen, dass die Seite geladen wird. Wenn Sie zu NetBeans zurückwechseln, sind Sie im Debug-Modus, und NetBeans hat die Ausführung am ersten Breakpoint unterbrochen. Nun können Sie beispielsweise Variableninhalte untersuchen, Schritt für Schritt den Programmablauf verfolgen und dadurch hoffentlich schneller Fehler finden.

Es ist empfehlenswert, die NetBeans-Einstellung *PHP → Debugging → Stop at First Line* zu deaktivieren, damit nur an den gesetzten Breakpoints gestoppt wird.

Schauen wir uns nun an, wie Sie Eclipse für das Debugging konfigurieren können. Gehen Sie zuerst in die Eclipse-Einstellungen, und stellen Sie sicher, dass in der Sektion *PHP → PHP Servers* ein Eintrag für *http://localhost* vorhanden ist. Unter *PHP → Debug* sollte der *PHP Debugger* auf *XDebug* gestellt sein, und bei *Server* sollte der eben eingerichtete Server für `localhost` angegeben sein. Auch hier empfehlen wir, die Einstellung *Break at First Line* zu deaktivieren, damit Eclipse nur an den gewünschten Breakpoints die Ausführung unterbricht.

Sie können im Quelltext Ihrer Anwendung einige Breakpoints setzen, indem Sie an den gewünschten Codeabschnitten doppelt in den linken Rand des Editorfensters klicken. Starten Sie den Debugger, indem Sie mit der rechten Maustaste auf die zu debuggende Datei klicken und dort *Debug as → PHP Web Page* auswählen. Im sich öffnenden Dialogfenster geben Sie die aufzurufende URL an, die Sie wie oben beschrieben finden können.

Nun wechselt Eclipse in die PHP-Debug-Ansicht, und Sie können mit dem Debugging Ihrer Anwendung beginnen. Auch hier können Sie den Inhalt von Variablen ansehen bzw. das Programm Zeile für Zeile ausführen. Wenn Sie das Debugging beenden wollen, müssen Sie dies durch Auswählen des Menüpunkts *Run → Terminate* machen.

Sie werden feststellen, dass Sie beim zweiten Starten des Debuggers nicht mehr nach der aufzurufenden URL gefragt werden. Dies liegt daran, dass Eclipse beim ersten Aufruf eine sogenannte *Debug Configuration* erstellt, in der (unter anderem) die aufzurufende URL gespeichert ist. Wenn Sie die aufzurufende URL ändern wollen, können Sie dies im Menü unter *Run → Debug Configurations* durchführen.

Debugging mit einer IDE ist eine schnelle und effiziente Möglichkeit, Fehler zu finden. Auch wenn die Debugger für PHP nicht so ausgereift sind wie ihre Pendants in Java, erleichtern sie doch die Fehlersuche erheblich.

Weitere hilfreiche Extensions und Links

Im Folgenden wollen wir Ihnen kurz einige hilfreiche Erweiterungen vorstellen, die zur Entwicklung von Extensions nützlich sind. Einige dieser Extensions (wie die ersten vier vorgestellten) werden auch vom Team um Extbase gepflegt, wogegen andere durch eigene Teams entwickelt werden.

blog_example

Das Blog-Beispiel wird im Verlauf des Buchs ausführlich erklärt. Es ist ein einführendes Beispiel, das viele Facetten und Features von Extbase und Fluid am Beispiel zeigt. Sie finden das Blog-Beispiel unter dem Extension-Key *blog_example* im TER oder auch unter *git://git.typo3.org/TYPO3v4/CoreProjects/MVC/blog_example.git* in Git.

viewhelpertest

Diese Extension demonstriert alle Fluid-ViewHelper und ihre Verwendung. Sie wird daher von den Fluid-Entwicklern zum Testen der Fluid-Distribution genutzt. Darüber hinaus ist sie auch hilfreich, um Praxisbeispiele zur Verwendung von ViewHelpern zu sehen. Sie finden die Extension im Git-Repository unter *git://git.typo3.org/TYPO3v4/CoreProjects/MVC/viewhelpertest.git*.

extension_builder

Der Extension Builder ermöglicht einen Schnelleinstieg in die Extbase-Extension-Programmierung, indem er viele wiederkehrende Aufgaben automatisiert. Die Verwendung des Extension Builder wird in Kapitel 4 dieses Buchs ausführlich erläutert.

Der Extension Builder ist im Git unter *git://git.typo3.org/TYPO3v4/Extensions/extension_builder.git* bzw. als stabile Version auch im TER unter dem Extension Key *extension_builder* zu finden.

devlog

Die Extension *devlog* (im TER verfügbar) kann zum Logging des Programmablaufs genutzt werden. Alle Log-Meldungen werden übersichtlich in einem Backend-Modul dargestellt. Die Extension ist besonders hilfreich, wenn in der Frontend-Ausgabe keine Debug-Meldungen ausgegeben werden sollen, wie beispielsweise auf Produktivsystemen.

Um Daten zu loggen, können Sie die Methode `\TYPO3\CMS\Core\Utility\GeneralUtility::devLog(*$message*, *$extensionKey*, *$severity*, *$data*)` nutzen. Dabei ist `$message` eine Nachicht, die geloggt werden soll, `$extensionKey` ist der Extension Key der aufrufenden

Extension, `$severity` ist eine Zahl von –1 bis 3, die den Schweregrad des Fehlers angibt, und `$data` ist ein optionales Array, das zusätzliche zu loggende Daten enthalten kann.

Weitere Informationen finden Sie in der Onlinedokumentation der Extension *devlog* im TER.

PHPUnit

Diese Erweiterung ermöglicht das Ausführen von automatisierten Unit-Tests im TYPO3-Backend. Sie finden diese Erweiterung im TER unter dem Extension Key *phpunit*. Auf die Verwendung dieser Extension wird noch ausführlich im Abschnitt »Test-Driven Development« in Kapitel 2, *Grundlagen*, eingegangen.

API-Dokumentation

Unter *http://api.typo3.org/extbase/current/* finden Sie zudem die jeweils aktuelle API-Referenz für Extbase. Analog dazu steht unter *http://api.typo3.org/fluid/current/* die entsprechende Fluid-Referenz zur Verfügung.

KAPITEL 2

Grundlagen

TYPO3 beeindruckt durch die große Anzahl der zur Verfügung stehenden Extensions. Wie bei Open Source-Projekten üblich, wurden diese Extensions von verschiedenen Programmierern geschrieben. Extensions kommen in den unterschiedlichsten Projekten zum Einsatz: Einige sind für den Privatgebrauch oder kleinere Vereine geschrieben, andere entstehen im Rahmen von Großprojekten von größeren Projektteams. Während ein Neuling, der gerade seine erste Extension schreibt, noch mit den Einstiegshürden von TYPO3 kämpft, werden bei großen Projekten oft hauseigene Frameworks eingesetzt. Dadurch sind der Stil und die Architektur heutiger Extensions sehr heterogen. Durch diese uneinheitliche Codebasis ist es oft sehr schwierig, vorhandene Extensions für eigene Projekte zu erweitern oder zu modifizieren, da man sich erst einmal mit der Denk- und Programmierweise des Autors bzw. des Autorenteams intensiv beschäftigen muss.

Mit Extbase soll unter anderem diese Varianz zwischen den Extensions reduziert werden: Bewährte Programmierparadigmen ermöglichen Einsteigern schnelle Erfolge und bewahren Entwickler davor, sich mit komplizierten Datenbankabfragen oder potenziellen Sicherheitslücken, wie SQL-Injections oder XSS-Attacken, herumschlagen zu müssen. Auf dieser Basis können sowohl kleinere Extensions als auch Großprojekte in einer strukturierten Art und Weise umgesetzt werden.

Extbase setzt auf vier ineinandergreifenden und sich ergänzenden Paradigmen auf, die wir Ihnen in diesem Kapitel vorstellen möchten. Diese begleiten Sie durch den gesamten Projektlebenszyklus, von der Planungsphase bis zur technischen Umsetzung und Pflege Ihrer Extension. Es handelt sich dabei um folgende Paradigmen:

- Objektorientierte Programmierung (OOP): Sie beschreibt, wie zusammengehörende Aspekte der realen Welt in abstrakte Objekte einer Software gekapselt werden können.
- Domain-Driven Design (DDD): Das Ziel dieses Entwicklungsansatzes ist es, die Begriffe, Regeln und Abläufe des zu lösenden Problems adäquat in Software umzusetzen.

- Model-View-Controller (MVC): Dieses Paradigma der Programmierung gibt eine saubere Trennung der Daten, der Ablaufsteuerung und der Ausgabelogik innerhalb einer Software vor.
- Test-Driven Development (TDD): Dieser Ansatz stellt die Basistechnik für stabilen, fehlerresistenten, lesbaren und damit wartbaren Code dar.

Jedes einzelne dieser vier Paradigmen ist in der professionellen Softwareentwicklung allgemein bekannt und mehr oder weniger weit verbreitet. Daraus ergibt sich ein gewichtiger Vorteil für die Anwendung von Extbase. Bisher war ein Experte für die Entwicklung von TYPO3-Extensions vor allem Experte in der Nutzung (und Umgehung) der Programmierschnittstellen (APIs) von TYPO3. Extbase fordert von Ihnen nun zusätzlich Kenntnisse und Fähigkeiten in Bereichen, die weit über die TYPO3-Community hinaus Gültigkeit und Wert besitzen. Damit steht Ihnen eine viel breitere Basis an Erfahrung in Form von Büchern, Foren oder persönlichen Kontakten zur Verfügung – ein nicht zu unterschätzender Aspekt für die Zukunft von TYPO3.

Kenntnisse in objektorientierter Programmierung, Domain-Driven Design und dem MVC-Paradigma sind für die Arbeit mit Extbase absolut notwendig. Kenntnisse in Test-Driven Development sind für das Verstehen oder die Verwendung von Extbase nicht unbedingt notwendig. Wir möchten Ihnen diese Entwicklungstechnik trotzdem wärmstens ans Herz legen.

Objektorientierte Programmierung mit PHP

Die objektorientierte Programmierung ist ein Programmierparadigma, das von Extbase und darauf aufbauenden Extensions vielfältig eingesetzt wird. In diesem Abschnitt geben wir Ihnen einen Überblick über die wichtigsten Konzepte der Objektorientierung.

Programme haben immer einen ganz speziellen Zweck, der – ganz allgemein gesprochen – darin besteht, ein Problem zu lösen. Mit »Problem« ist nicht unbedingt ein Fehler oder ein Defekt gemeint, sondern vielmehr eine reale Aufgabe. Dieses Problem hat meist eine ganz konkrete Entsprechung im richtigen Leben.

Beispielsweise könnte ein Programm sich des Problems annehmen, die Buchung einer Kreuzfahrt im Indischen Ozean auszuführen. Wir haben hier offensichtlich ein Problem (nämlich, dass der buchende Programmierer viel zu viel arbeitet und nun endlich in den Urlaub fahren will) und ein Programm, das Erholung verspricht, indem es eine Kabine auf einem der Luxusliner für ihn und seine Frau reservieren könnte.

Bei der Objektorientierung wird angenommen, dass man konkrete Probleme mit Programmen lösen will, und konkrete Probleme werden prinzipiell von realen Objekten ausgelöst. Daher steht hier das Objekt im Mittelpunkt. Dies kann natürlich auch abstrahiert sein – nicht immer kann man etwas derart Konkretes wie ein Auto oder ein Schiff damit abbilden, sondern muss natürlich auch eine Reservierung, ein Konto oder ein grafisches Symbol darstellen.

Objekte sind »Behälter« für Daten und dazugehörige Funktionalität. Die Daten eines Objekts werden in dessen *Eigenschaften* (*Properties*) gespeichert. Die Funktionalität wird durch *Methoden* bereitgestellt, welche beispielsweise die Eigenschaften des Objekts verändern können. Bezogen auf das Kreuzfahrtschiff lässt sich beispielsweise sagen, dass es eine bestimmte Anzahl an Kabinen, eine Länge und Breite und eine Höchstgeschwindigkeit hat. Dies sind alles Eigenschaften dieses Schiffs. Außerdem hat es Methoden, um den Motor zu starten (und hoffentlich auch wieder stoppen), die Richtung zu ändern sowie um mehr Schub zu geben, damit man sein Urlaubsziel etwas schneller erreicht.

Warum überhaupt Objektorientierung?

Nun wird sich sicher der eine oder andere Leser fragen, warum denn nun überhaupt objektorientiert programmiert werden soll und nicht – wie bisher – etwa prozedural, also schlicht mit einer Aneinanderreihung von Funktionen. Schaut man sich die knapp 4.300 Extensions an, die es für TYPO3 mittlerweile gibt, stellt man fest, dass diese zwar standardmäßig mit einer Klasse ausgestattet wurden – vom Extension-Programmierer aber in vielleicht 95 % aller Fälle prozedural weiterprogrammiert wurden.

Die prozedurale Programmierung hat aber einige schwerwiegende Nachteile:

- Inhaltlich zusammengehörende Eigenschaften und Methoden können nicht zusammengefasst werden. Diese in der Objektorientierung als *Kapselung* bezeichnete Methodik ist aber schon allein für die Übersicht sehr wichtig.
- Einmal geschriebener Code kann relativ schlecht wiederverwendet werden.
- Alle Eigenschaften können an jeder beliebigen Stelle verändert werden. Dadurch entstehen schnell Fehler, die sich nur sehr schwer identifizieren lassen.
- Prozeduraler Code wird sehr schnell unübersichtlich. Dies bezeichnet man als Spaghetti-Code.

Die Objektorientierung ist zudem viel näher dran an der realen Welt, in der es reale Objekte gibt, die allesamt Eigenschaften und (meist) auch Methoden haben. Dieser Umstand wird nun auch in der Programmierung berücksichtigt.

Es soll nun im Folgenden grundsätzlich um das Objekt »Schiff« gehen. Wir wollen dieses Objekt überhaupt erst einmal zum Leben erwecken, es mit Kabinen, einem Motor und weiteren nützlichen Dingen ausstatten. Zudem wird es Funktionen geben, die das Schiff bewegen, also den Motor an- und ausschalten. Später werden wir sogar einen Luxusliner auf Basis des normalen Schiffs kreieren und diesen sogar mit einem Golf-Simulator und Satelliten-TV ausstatten.

Auf den folgenden Seiten wollen wir Ihnen die objektorientierte Programmierung so plastisch wie möglich (aber dennoch immer systematisch korrekt) nahebringen. Und dies hat auch einen ganz speziellen Grund: Je mehr Sie sich mit dem Objekt und dessen Eigenschaften und Methoden identifizieren können, desto offener sind Sie für die hinter OOP

stehende Theorie. Beides ist für eine erfolgreiche Programmierung notwendig – wenn auch die Objekte, die Ihnen später begegnen werden, oftmals nicht so deutlich imaginiert werden können.

Klassen und Objekte

Wir gehen jetzt noch einmal einen Schritt zurück und stellen uns vor, dass es eine Art Bauplan für Schiffe im Allgemeinen gibt. Nun interessiert uns erst einmal nicht das konkrete Schiff, sondern die Blaupause dafür. Diese wird als *Klasse* bezeichnet, in diesem Fall ist es die Klasse `Ship`. In PHP wird dies wie folgt notiert:

```
<?php

class Ship {

   ...

}

?>
```

In diesem Codeabschnitt haben wir noch ordnungsgemäß die notwendigen PHP-Tags am Anfang und Ende notiert. Diese werden wir bei den folgenden Beispielen weglassen, um die Listings etwas kürzer zu halten.

Mit dem Schlüsselwort `class` wird die Klasse also eingeleitet, und innerhalb der geschweiften Klammern werden Eigenschaften und Methoden notiert. Diese Eigenschaften und Methoden wollen wir nun ergänzen:

```
class Ship {

   public $name;
   public $coaches;
   public $engineStatus;
   public $speed;

   function startEngine() {}
   function stopEngine() {}
   function moveTo($location) {}

}
```

Unser Schiff hat nun einen Namen (`$name`), eine Anzahl an Kabinen (`$coaches`) und eine Geschwindigkeit (`$speed`). Zusätzlich haben wir eine Variable eingebaut, die den Zustand der Motoren beinhaltet (`$engineStatus`). Selbstverständlich besitzt ein Schiff viel mehr Eigenschaften, die auch alle irgendwie wichtig sind – für unsere Abstraktion reicht diese Menge an Eigenschaften zunächst aber aus. Warum vor den Eigenschaften das Schlüsselwort `public` steht, wird uns weiter unten beschäftigen.