Programmierung, Anpassung & Integration

Magento

Das Handbuch für Entwickler

Roman Zenner & Vinai Kopp

sowie Claus Nortmann, Sebastian Heuer, Dimitri Gatowski & Daniel Brylla von Visions new media

O'REILLY®

Magento

Das Handbuch für Entwickler

Roman Zenner, Vinai Kopp
sowie Claus Nortmann, Sebastian Heuer,
Dimitri Gatowski & Daniel Brylla
von Visions new media

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

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

1. Auflage 2010

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

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

ISBN 978-3-89721-928-1

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Einführung

Die E-Commerce-Software Magento bleibt eine Erfolgsstory: Die Anzahl der Downloads steigt, die Community wächst und gedeiht, und zunehmend ziehen auch größere Unternehmen Magento als Alternative zu etablierten kommerziellen Softwareprodukten in Betracht. Dieses verstärkte Interesse an dem Shopsystem ist nicht zuletzt der Magento Enterprise Edition (EE) und Professional Edition (PE) zu verdanken, die dem bereits erstaunlichen Funktionsumfang der Community Edition (CE) das Sahnehäubchen aufsetzten.

Magento bietet durch seinen modularen Aufbau und sein flexibles Datenmodell Shopbetreibern die Möglichkeit, komfortabel individuelle Prozesse abzubilden und so die Software den Geschäftsprozessen anzupassen. Durch das Magento-Baukastenprinzip lassen sich eigens programmierte Erweiterungen einfach in die Applikation einfügen, ohne Systemdaten zu überschreiben und damit die Updatefähigkeit des Systems zu beeinträchtigen. Darüber hinaus stehen auf dem Extension-Marktplatz *MagentoConnect* bereits über 2.000 fertige Erweiterungen zur Verfügung, mit denen Sie die Funktionen in Ihrem Shop nachrüsten können, die möglicherweise nach der Standardinstallation fehlen.

Aufgrund des nachhaltigen Erfolgs des Systems wird der Ruf nach erfahrenen Magento-Entwicklern immer lauter. Die zurzeit noch relativ überschaubare Anzahl von Dienstleistern in diesem Bereich kommt den Magento-Projektanfragen kaum noch hinterher, längere Wartezeiten sind längst an der Tagesordnung. Umso wichtiger ist es, Entwicklern, die sich mit Magento auseinandersetzen und es für die eigenen Zwecke einsetzen möchten, die entsprechenden Hilfsmittel an die Hand zu geben. Zwar werden im offiziellen Forum bereits seit Beginn technisch detaillierte Probleme diskutiert, und in den einschlägigen Fachblogs nehmen die Autoren Entwicklerthemen in Angriff, wünschenswert wäre allerdings eine erschöpfende Dokumenation des strukturellen Aufbaus von Magento und der einzelnen Klassen, aus denen die Software besteht. Obwohl dies seit Ende 2009 in Angriff genommen wird, fehlt bislang die nötige technische Fachliteratur zum Thema Magento.

Das vorliegende Buch soll diese Lücke schließen. Es wurde von sechs erfahrenen »Magentoianern« verfasst, die diese Software schon in zahlreichen Projekten eingesetzt und somit Magento bis in den letzten Winkel kennengelernt haben. Insbesondere im zweiten Teil des Buchs, der sich mit der praktischen Arbeit mit dem Shopsystem beschäftigt, lassen wir Erfahrungen aus dem Agenturalltag einfließen und stellen Probleme sowie deren Lösungen vor, die für viele Entwickler relevant sein dürften.

An wen sich dieses Buch richtet

Dieses Buch richtet sich vornehmlich an Entwickler, die bereits Erfahrungen mit objektorientierter Programmierung (OOP) mit PHP sammeln konnten. Da Magento in PHP geschrieben ist und auf dem Zend Framework basiert, haben all diejenigen die Nase vorn, die mit dieser Konstellation schon gearbeitet haben. Die Erfahrung hat jedoch gezeigt, dass dies kein Muss ist: Oft haben es beispielsweise gestandene Java-Programmierer wesentlich einfacher, sich im Magento-Umfeld zurechtzufinden als PHP-Entwickler, die diese Sprache bis dato eher prozedural (Spaghetti-Code) genutzt haben und sich nun völlig anderen Strukturen gegenübersehen.

In den nachfolgenden Kapiteln werden auch all diejenigen, die sich eher mit der Optik eines neuen Magento-Shops beschäftigen, wichtige Hintergrundinformationen zur Arbeit mit dem Magento-Frontend erhalten. Wenn Sie Gestalter oder Designer sind und Ihnen der eine oder andere PHP-Tag im bekannten HTML-Code kein Gräuel ist, werden Sie dieses Buch ebenfalls hilfreich finden.

Beim Verfassen der einzelnen Kapitel sind wir davon ausgegangen, dass die wesentlichen Funktionalitäten von Magento schon bekannt sind und wir bei Ihnen bereits auf Verständnis stoßen, wenn wir Begrifflichkeiten wie *Admin-Panel/Backend*, *MagentoConnect* oder *Systemkonfiguration* in den Ring werfen. Um sich über diese Grundlagen einen Überblick zu verschaffen, sei Ihnen wärmstens das Buch *Online-Shops mit Magento* (O'Reilly Verlag 2009) ans Herz gelegt, das von einem Mitglied dieses Autorenteams geschrieben wurde.

Aufbau dieses Buchs

Dieses Buch ist in zwei Teile unterteilt: Im ersten Teil gehen wir auf die Magento-Architektur ein und beleuchten im Detail die (Programmier-)Konzepte, die Magento zugrunde liegen. Der zweite Teil des Buchs ist anwendungsorientierter und vermittelt anhand vieler kochbuchartiger Rezepte, wie Sie Magento in der Praxis nutzen. Dort erfahren Sie, wie Sie mit Magento eigene Extensions entwickeln und somit individuelle Funktionalitäten realisieren können.

Ziel dieses Buches ist es, dass Sie die wesentlichen Bestandteile des Systems und deren Verknüpfung verstehen und somit in der Lage sind, neue Anforderungen beispielsweise

im Rahmen einer eigenen Extension umzusetzen. Die Rezepte aus dem zweiten Teil des Buchs sollen Ihnen dabei Lösungsansätze für Probleme aufzeigen, die Ihnen in der täglichen Arbeit mit Magento so oder so ähnlich sicherlich über den Weg laufen werden.

In den Praxisbeispielen und Rezepten dieses Buchs wird Ihnen öfter der mysteriöse *Webkochshop* begegnen: Hierbei handelt es sich um einen fiktiven Onlineshop, den wir bereits im Buch *Online-Shops mit Magento* zur Verdeutlichung von Beispielen verwendet haben. Wir sind davon überzeugt, dass dieser Anbieter von Küchenausstattung und feinen Koch- und Backzutaten uns auch in diesem Buch wertvolle Dienste leisten wird.

Im Folgenden möchten wir auf die Themen der einzelnen Kapitel etwas genauer eingehen:

In Kapitel 1, *Der erste Eindruck*, geben wir einen ersten Überblick über die Magento-Software. Dabei gehen wir zunächst auf das Zend Framework als das zugrunde liegende PHP-Framework ein und diskutieren, welche Programmierstrategien und -konventionen Magento von diesem Framework geerbt hat. Anschließend sehen wir uns Magentos modularen Aufbau und die strikte Trennung von Gestaltungs- und Geschäftslogik an und erläutern, wie sich das in der Datei- und Verzeichnisstruktur des Systems niederschlägt. Zum Schluss gehen wir näher auf das MVC-Pattern als Ordnungsprinzip und seine Realisierung im Magento-System ein.

Kapitel 2, *Eigene Extensions entwickeln*, widmet sich im Detail der Programmierung von Magento-Extensions. Sie lernen, wie eine Extension aufgebaut ist und welche obligatorischen Bestandteile erstellt bzw. konfiguriert werden müssen. Anschließend zeigen wir Ihnen verschiedene Strategien, vorhandene Magento-Klassen so zu erweitern oder zu überlagern, dass die von Ihnen geplante Geschäftslogik vom System übernommen wird, ohne tatsächlich Dateien des Kernsystems zu überschreiben. Wir beschließen das Kapitel mit der Frage, wie sich mithilfe von sogenannten Installations- und Update-Skripten Änderungen an der Magento-Datenbank durchführen lassen, die für das Funktionieren Ihrer neuen Extension notwendig sind.

Kapitel 3, *Models und Resource-Models*, steht ganz im Zeichen der Datenhaltung und -bearbeitung. Wir diskutieren die Funktion von Models und Resource-Models und gehen auf das EAV-Modell und seine Nutzung innerhalb von Magento ein. Außerdem untersuchen wir, wie Daten mithilfe sogenannter Collections sortiert und gefiltert werden können, und beenden das Kapitel mit einem Praxisbeispiel, das das Zusammenspiel von Models und Resource-Models veranschaulicht.

Nachdem wir uns in den ersten Kapiteln vornehmlich mit der oftmals nicht sichtbaren Hintergrundstrukur beschäftigt haben, dreht sich in Kapitel 4, *Das Magento-Frontend*, alles um die Benutzeroberfläche des Shopsystems. Wir erklären, was man im Magento-Universum unter Blöcken, Layouts und Templates versteht, erläutern die Bedeutung von Themes und Packages und zeigen anschließend, wie diese Bestandteile zusammenspielen.

Kapitel 5, *Produkte und Kategorien*, läutet den praktischen Teil des Buchs ein. An dieser Stelle haben wir eine Reihe von Rezepten zusammengestellt, die sich konkret mit der Anzeige von Produkten und Kategorien beschäftigen. So diskutieren wir beispielsweise die Erstellung einer vertikalen Navigation oder die Anzeige von zusätzlichen Attributen in der Produktauflistung.

In Kapitel 6, *Angebote und Bestellungen*, stellen wir einen weiteren zentralen Bereich von Magento vor. Wenn Sie wissen möchten, wie man einen neuen Status für Bestellungen einrichtet, Bestelldaten durch zusätzliche Attribute anreichert oder einen produktspezifischen Versandaufpreis realisiert, sind Sie hier an der richtigen Stelle.

Das Ziel von Kapitel 7, *Systemintegration*, ist es, Ihnen den Datenaustausch zwischen einem Drittsystem und Magento näherzubringen. Die Rezepte in diesem Kapitel beschäftigen sich beispielsweise mit der Frage, wie Produktbestände synchronisiert oder Aufträge an ERP-Systeme gesendet werden können.

Magentos Geschwindigkeit ist Thema von Kapitel 8, *Performance und Skalierbarkeit*. Die hier gesammelten Rezepte beschäftigen sich unter anderem mit der Frage, wie sich die Performance von Magento mittels Caching verbessern lässt und wie man einfache Lasttests durchführen kann.

Auch in Kapitel 9, *Deployment*, geht es vor allem um die Realisierung von konkreten Projekten. Wir erläutern im Detail das Deployment eines Magento-Shops und gehen auf Release-Management sowie auf Versionierung ein.

Kapitel 10, *Bezahlung und Versand*, beschäftigt sich mit einer weiteren zentralen Säule von Magento. Alle diejenigen, die eine neue Versand- oder Bezahlart integrieren möchten, finden in den Rezepten dieses Kapitels sicherlich wertvolle Hinweise.

In Kapitel 11, *Das Admin-Panel erweitern*, finden Sie Rezepte, die sich beispielsweise der Frage annehmen, wie sich ein neues Admin-Grid einbinden und gestalten lässt und was Sie benötigen, um bestimmte Skripte Ihrer Extensions mithilfe von Cronjobs automatisiert in regelmäßigen Abständen ausführen zu lassen.

Der Anhang beinhaltet eine Referenz der Attributeigenschaften von Magento. Darüber hinaus finden Sie hier eine Übersicht über die Payment-API von Magento.

Typografische Konventionen

In diesem Buch gelten folgende typografische Konventionen:

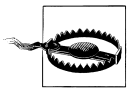
Kursivschrift
: Kennzeichnet neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateierweiterungen.

`Nichtproportionalschrift`
: Wird für Programmlistings sowie innerhalb von Absätzen zum Kennzeichnen von Programmelementen wie Variablen-, Klassen- und Funktionsnamen, Datenbanken, Datentypen, Anweisungen und Schlüsselwörtern verwendet.

`Nichtproportionalschrift fett`
: Zeigt Befehle oder anderen Text, der unverändert vom Benutzer eingegeben werden muss.

`Nichtproportionalschrift kursiv`
: Zeigt Text, der durch benutzereigene Werte oder Werte ersetzt werden soll, die sich aus dem Kontext ergeben.

Dieses Symbol kennzeichnet einen Tipp, einen Vorschlag, einen nützlichen Hinweis oder eine allgemeine Anmerkung.



Dieses Symbol kennzeichnet eine Warnung oder einen Aufruf zur Vorsicht.

Codebeispiele

Besonders im zweiten Teil des Buchs verwenden wir eine Reihe von Codebeispielen, mit deren Hilfe wir die Funktionen und Zusammenhänge der Magento-Programmierung erläutern möchten. Falls Sie selbst diese verwenden möchten, müssen Sie sie nicht umständlich abtippen, sondern können Sie bequem von der Verlagswebsite unter *http://examples.oreilly.de/german_examples/magentopaiger/* herunterladen. Tabelle E-1 zeigt eine Übersicht darüber, welche Module in welchen Kapiteln zu finden sind.

Tabelle E-1: Codebeispiele zum Download

Kapitel	Name des Codearchivs
Kapitel 2	Webkochshop_HelloWorld-0.1.0.zip
	Webkochshop_OrderAlert-0.1.0.zip
Kapitel 3	Webkochshop_Rezepte-0.1.0.zip
Kapitel 4	Webkochshop_LayoutUpdates-0.1.0.zip
Kapitel 5	Webkochshop_Navigation-0.1.0.zip
	Webkochshop_Kategorieansicht-0.1.0.zip
	Webkochshop_Vergleichsliste-0.1.0.zip
	Webkochshop_Kundenpreise-0.1.0.zip
	Webkochshop_ProductWidget-0.1.0.zip
Kapitel 6	Webkochshop_QualityAssurance-0.1.0.zi
	Webkochshop_Versandaufpreis-0.1.0.zip
	Webkochshop_GratisArtikel-0.1.0.zip
	Webkochshop_OrderKommentar-0.1.0.zip
	Webkochshop_Import-0.1.0.zip

Tabelle E-1: Codebeispiele zum Download (Fortsetzung)

Kapitel	Name des Codearchivs
	Webkochshop_SOAP.zip
Kapitel 7	Webkochshop_OrderExport-0.1.0.zip
Kapitel 10	Webkochshop_Shipping-0.1.0.zip
	Webkochshop_Payment-0.1.0.zip
Kapitel 11	Webkochshop_Lieferant-0.1.2.zip

Website zum Buch

Die Webseite zum Buch finden Sie unter *www.mage-buch.de*. Dort finden Sie nicht nur alle Listings zum Download, sondern auch einen Entwickler-Blog, weiterführende Links und Errata zum Buch. Wir freuen uns auf Ihren Besuch!

Wichtige Vorbereitungen treffen

Bevor Sie zu Ihren ersten Magento-Abenteuern aufbrechen, sollten Sie auf ein laufendes Magento-Testsystem zugreifen und sich die einzelnen Skripte der Applikation mithilfe einer passenden IDE auch ansehen können. In diesem Abschnitt gehen wir auf die wichtigsten vorbereitenden Maßnahmen ein.

Das Testsystem installieren

Für die Betrachtung und nähere Erläuterung des Systems und das Nachvollziehen der Codebeispiele gehen wir davon aus, dass Sie Zugriff auf eine installierte Version von Magento haben und sich sowohl die Programmdateien als auch die zugrunde liegende Datenbank ansehen können.

Als Grundlage sowohl für die Ausführungen im Architekturteil wie auch für die Rezepte dient uns die Magento-Community-Edition (CE) in der Version 1.4, die seit Ende Februar 2010 als Download verfügbar ist.

Damit während des Entwickelns Fehler angezeigt werden, muss entweder die Umgebungsvariable `MAGE_IS_DEVELOPER_MODE` in der Apache-Konfiguration für den Host gesetzt werden (mit der Anweisung `SetEnv MAGE_IS_DEVELOPER_MODE 1`), oder in der *index.php* der Aufruf `Mage::setIsDeveloperMode(true)`; aus der `if`-Anweisung entfernt werden, sodass er immer aktiv ist. Diese Variante sollte jedoch vor dem Launch des Shops wieder rückgängig gemacht werden!

Zusätzlich ist es sinnvoll, während der Entwicklung das Kommentarzeichen vor dem Befehl `ini_set('display_errors', 1);` zu entfernen, damit auch schwere PHP-Fehler angezeigt werden.

Die Entwicklungsumgebung einrichten

Es gibt eine Vielzahl von IDEs, mit denen Sie sich die Arbeit mit Magento erleichtern können. Für welche genau Sie sich entscheiden, bleibt natürlich Ihnen überlassen. Sie sollten nur darauf achten, dass ein gut funktionierendes Syntax-Highlighting und ein schneller und übersichtlicher Klassenbrowser vorhanden sind; besonders Letzterer wird Ihnen bei den vielen Klassen und Methoden der Magento-Applikation nützliche Dienste leisten. Ohne Anspruch auf Vollständigkeit haben wir eine kleine Liste von IDEs zusammengestellt, mit denen wir in der Vergangenheit bereits gute Erfahrungen haben sammeln können:

- Zend Studio *(http://shop.zend.com/de/zend-studio-for-eclipse.html)*
- Netbeans *(http:netbeans.org)*
- Eclipse *(http://eclipse.org)*
- Komodo *(http://www.activestate.com/komodo/)*
- Aptana *(http://aptana.org)*

Neben diesen IDEs ist es besonders für die Frontend-Entwicklung enorm hilfreich, auf das *Firebug*-Plug-in für den *Firefox*-Browser zurückgreifen zu können. Damit lassen sich beispielsweise lokale Änderungen im (X)HTML- und CSS-Code vornehmen, deren Ergebnisse sofort angezeigt werden. Ebenso wichtig ist der integrierte JavaScript-Debugger für alle diejenigen, die an den JavaScript-Dateien Ihres Shopprojekts Änderungen vornehmen wollen.

In diesem Zusammenhang hat die Firma *netresearch* aus Leipzig eine kostenlose Magento-Extension via MagentoConnect veröffentlicht, die den Debugging-Output von Magento optisch aufbereitet und ihn in Firebug darstellt. Weitere Informationen erhalten Sie unter *http://www.magentocommerce.com/extension/949/*.

Eine Versionskontrolle nutzen

Gerade bei größeren Projekten, bei denen mehrere Entwickler gleichzeitig an Erweiterungen und Modifikationen arbeiten, ist es unbedingt empfehlenswert, mit einem Versionierungssystem zu arbeiten. Aber auch als Einzelkämpfer kann es mitunter viel Frust und Zeit sparen, in der Lage zu sein, nach einer glücklosen Codeänderung zu einer älteren, noch funktionierenden Programmversion zurückkehren zu können.

In diesem Zusammenhang haben die Autoren dieses Buchs sehr gute Erfahrungen mit *Subversion* (*SVN*) und dem moderneren *GIT* machen können. Mit diesem System lassen sich Code-Trunks komfortabel verwalten, außerdem verfügen die oben erwähnten IDEs bereits über entsprechende Plug-ins, sodass nicht extra ein SVN- oder GIT-Client verwendet werden muss und der gewohnte Workflow beibehalten werden kann.

Als Best Practice bei der Verwendung von Versionskontrollsystemen im Magento-Kontext hat sich erwiesen, für jedes neue Projekt einen neuen *Branch* zu öffnen und einen

separaten *Vendor Branch* vorzuhalten, der die neuen Magento-Releases enthält. Auf diese Weise können Sie ohne großen Aufwand via *Merge* ein aktuelles Projekt auf den neuesten Stand bringen, wenn eine neue Version von Magento veröffentlicht wird. Weitere Informationen zu GIT, SVN und deren Möglichkeiten finden Sie unter *http://git-scm.com/* und *http://subversion.tigris.org/*.

Wir stellen vor: Die Community

Wenn Sie sich eine Weile mit Magento beschäftigt haben, verfügen Sie über das nötige Rüstzeug, um Erweiterungen für Ihre eigenen Bedürfnisse zu programmieren. Trotzdem gibt es ab und an einen Punkt, an dem Sie möglicherweise feststecken und einen kleinen Hinweis zum weiteren Vorgehen gut gebrauchen könnten. In diesem Zusammenhang ist es gut zu wissen, dass hinter Magento eine stetig wachsende internationale Community steht, die die verschiedensten Probleme aktiv diskutiert und beispielsweise im offziellen Magento-Forum (*http://www.magentocommerce.com/boards/*) Hilfestellung leistet. Aber auch andersherum wird ein Schuh daraus: Wenn Sie eine schöne Lösung oder einen eleganten Ansatz zu einem Problem gefunden haben und davon überzeugt sind, dass dieser Geistesblitz auch anderen Magento-Entwicklern helfen könnte, bloggen Sie vielleicht darüber oder schreiben einen Wiki-Eintrag (*http://www.magentocommerce.com/wiki/*). Oder Sie schauen ab und zu mal ins Forum und beantworten eine Frage, die noch unbeantwortet im Raum steht. Magento ist eine Open Source-Software und basiert per definitionem auf Geben und Nehmen und dem freiwilligen Austausch von Informationen. Zwar steht mit der Firma Varien eine erfahrene E-Commerce-Agentur hinter Magento, die bei der Entwicklung der Software den Hut aufhat sowie professionelle Dienstleistungen rund um die Software anbietet – letzten Endes ist es aber die Community, die solch eine Software erst zu dem macht, was sie ist. Ein gutes Beispiel hierfür ist die in »Magento Contributor Agreement«, auf Seite XVII beschriebene Möglichkeit, selbst Patches zu Magento beizusteuern und sich damit aktiv an der Weiterentwicklung zu beteiligen. Bevor wir auf die wichtigsten Vorgänge innerhalb der Magento-Community eingehen, möchten wir kurz das *Community Advisory Board* vorstellen, das gegenüber Varien als Sprachrohr der Community dienen und sozusagen eine Brücke zwischen den vielen Magento-Enthusiasten weltweit und den Core-Entwicklern bei Varien schlagen soll.

Das *Community Advisory Board* (*CAB*) wurde Ende April 2009 ins Leben gerufen. Es besteht unter anderem aus den Community-Managern verschiedener Länder (zurzeit sind dies Deutschland, die Niederlande und Frankreich), dem CEO von Varien, Roy Rubin und dem offiziellen Community-Manager Koby Oz, der ebenfalls bei Varien angestellt ist. Das CAB kommt in regelmäßigen Abständen zusammen, um die weiteren Entwicklungen von Magento abzustimmen und zwischen der weltweiten Community und den Magento-Entwicklern zu vermitteln. Die nun folgenden Bereiche der Community-Arbeit sind im Wesentlichen auf Initiativen des CAB zurückzuführen.

Magento Contributor Agreement

Seit Oktober 2009 haben Entwickler offiziell die Möglichkeit, eigene Codebeiträge, d.h. Bugfixes und Funktionserweiterungen bzw. -verbesserungen in Form von Patches, an Varien zu schicken. Diese werden dort technisch und inhaltlich geprüft und finden, wenn sie für gut befunden werden, ihren Weg in den offiziellen Core. Die rechtliche Basis für diese Art der Mitwirkung ist die Unterzeichnung des sogenannten *Magento Contributor Agreement* (*http://www.magentocommerce.com/images/uploads/MCA-Magento-Contributor-Agreement-230909.pdf*). Dieses Dokument stellt einen Vertrag dar, mit dem Sie zum einen bestätigen, alleiniger Autor dieser Beiträge zu sein und keine Rechte Dritter zu verletzen. Außerdem übertragen Sie der Firma Varien sämtliche Rechte an Ihren Entwicklungen und erklären, in diesem Zusammenhang auch keine weiteren Forderungen geltend zu machen. Nachdem Sie das unterschriebene Dokument per Fax oder E-Mail an Varien geschickt haben, können Ihre Beiträge in jeder erdenklichen Form weiterverwendet werden, z.B. als fester Bestandteil des Programmkerns oder auch in kommerziell vertriebenen Programmteilen. So stellt Varien langfristig sicher, dass nicht irgendwann einmal die Situation auftritt, dass arbeits- und zeitintensiv weite Teile des Cores geändert werden müssen, weil ein Entwickler nachträgliche Forderungen stellt.

Zur Diskussion über diese Einreichungsvorgänge und Code-Patches wurde eine eigene Google-Gruppe eingerichtet. In der *Magento Development Google Group* (*http://groups.google.com/group/magento-devel*) sind aktuell 180 Mitglieder aus der ganzen Welt angemeldet und diskutieren sinnvolle Erweiterungen des Magento-Cores.

Magento Community Documentation

Ein viel zitiertes Manko der täglichen Arbeit mit Magento ist das Fehlen einer offiziellen Dokumentation. Im Gegensatz zu beispielsweise dem Zend Framework, zu dem ein mehrere Hundert Seiten starkes Dokument existiert, das die einzelnen Komponenten im Detail erklärt, findet man außer einer via *phpdoc* (*http://docs.magentocommerce.com/*) automatisch generierten Dokumentation bei Magento nichts Vergleichbares. Die Erfahrung hat zwar gezeigt, dass viele wichtige Informationen für Entwickler in Forenbeiträgen, Wiki-Artikeln, Webinars und Screencasts enthalten sind, die Recherche kann aber genau aus diesem Grund teilweise sehr aufwendig werden.

Zum Ende 2009 hat man sich entschlossen, die Entwicklung einer solchen ausführlichen Dokumentation in geregelte Bahnen zu lenken und in Person von Matt Blackwell das weiter oben bereits erwähnte Wiki mithilfe der Community neu zu strukturieren bzw. zu erweitern. Analog zu den Codebeiträgen (siehe »Magento Contributor Agreement«, auf Seite XVII) unterliegt auch das Einsenden von Dokumenationsteilen dem *Magento Contributor Agreement*, d.h. eigene Wiki-Artikel, die beispielsweise die Funktionen bestimmter Klassen und Methoden im Detail erläutern, werden ohne weitere Forderungen an Varien übergeben.

Magento Community Edition Roadmap

Wer wissen möchte, wohin die Magento-Reise noch gehen wird und welche Funktionen diese Software in Zukunft haben wird, dem sei die *Magento Community Edition Roadmap* (*http://magento.uservoice.com/forums/24441-magento-community-edition-roadmap*) ans Herz gelegt. Dahinter verbirgt sich eine Uservoice-Seite, über die jeder seine Wünsche für zukünftige Magento-Versionen äußern und diese zur gleichen Zeit auch zur Abstimmung freigeben kann. Seit August 2009 kann jedes Mitglied der Community mit seinen Ideen und seinen Stimmen Magentos Entwicklungsrichtung beeinflussen kann (siehe Abbildung E-1).

Abbildung E-1: Die Magento Community Edition Roadmap bei Uservoice

Zurzeit sind dort gut 270 Themen bzw. Wünsche eingestellt, von denen für den Punkt »Optimize the code« gut 1.700 Stimmen abgegeben wurden. Weitere populäre Einträge zielen auf die Bearbeitbarkeit einer Bestellung (»Edit order without creating a new one) und die oben bereits angesprochene Dokumentation (»Document the system properly«).

Es ist geplant, dass das CAB monatlich die wichtigsten bzw. spannendsten Themen mit dem Magento-Team bespricht, um herauszufinden, ob und in welchem Zeitraum diese implementiert werden können. Dieser Prozess soll dann in eine »offizielle« Roadmap münden, die auf der Magento-Website veröffentlicht wird.

Weitere Quellen

Sollte doch einmal eine Frage offenbleiben, haben wir weiteres Online- und Offlinematerial kurz für Sie aufgeführt, das Ihnen bei der Arbeit mit Magento zusätzlich Hilfestellung geben kann.

Literatur

Wenn Sie sich noch stärker mit den technischen Grundlagen von Magento selbst beschäftigen möchten, seien Ihnen folgende Bücher ans Herz gelegt:

- Hilfreiche Rezepte rund um die objektorientierte Programmierung mithilfe von PHP 5: *PHP5-Kochbuch* von David Sklar et al. (O'Reilly 2009)
- Eine umfangreiche Einführung in das Zend Framework: *Das Zend Framework* von Ralf Eggert (Addison-Wesley 2009)
- Das Handbuch rund um das Thema JavaScript: *JavaScript – Das umfassende Referenzwerk* von David Flanagan (O'Reilly 2007)

Weblinks

Mittlerweile hat sich eine bunte Mischung von technischen und nicht technischen Blogs, Screencasts, Podcasts zum Thema Magento etabliert, von denen hier stellvertretend einige genannt werden sollen.

Magento-Blog
: Das offizielle Magento-Blog (*magentocommerce.com/blog*) informiert über neueste Entwicklungen aufseiten Variens und der Community.

Magentofeeds
: Über *Magentofeeds.com* wird eine ganze Reihe von Blogs gesammelt, die sich mit dem Thema Magento auseinandersetzen. Wenn Sie diesen RSS-Feed abonnieren, entgehen Ihnen mit Sicherheit keine Neuigkeiten aus dem Magento-Universum mehr.

Magento-Podcast
: Dieser Podcast (*magentopodcast.de*) wird von Rico Neitzel und Roman Zenner regelmäßig produziert und setzt sich mit Schwerpunktthemen aus der Magento-Welt auseinander.

Veranstaltungen
: Inzwischen haben sich einige regelmäßige Veranstaltungen etabliert, bei denen sich Magento-Interessierte und -Entwickler treffen und in persona austauschen können. Neben diesem Austausch werden in Vorträgen und Workshops verschiedene Aspekte der Shoperstellung auf Basis von Magento erörtert und diskutiert. Zu nennen sind hier vor allem:

  - Meet-Magento Deutschland (*meet-magento.de*)
  - Meet-Magento Niederlande (*meet-magento.nl*)
  - Bargento Frankreich (*bargento.fr*)

Danksagungen

Roman

Ich möchte mich bei meinen Co-Autoren, besonders aber bei Vinai Kopp für den kreativen Austausch und die interessanten Diskussionen rund um Magento bedanken. Ein weiteres Dankeschön geht an unsere Lektorin Inken Kiupel, die uns stets mit Rat und Tat zur Seite gestanden hat. Last – but certainly not least – möchte ich mich bei meiner Frau bedanken, die mich so geduldig an Abenden und Wochenenden entbehrt hat.

Vinai

Mein Dank geht an die Magento-Entwickler-Community für all die Unterstützung. Open Source-Software mit einer lebendigen Community ist das Herz des Internets. Außerdem danke an Rico Neizel und Roman Zenner, ich freue mich auf zukünftige gemeinsame Abenteuer! Danke an meine Frau und meine Kinder, für alles!

Claus, Sebastian, Dimitri und Daniel

Nachdem wir die Entwürfe einiger Kapitel erarbeitet hatten, stießen Roman und Vinai zum Autorenteam hinzu – wir bedanken uns sehr herzlich für ihren zupackenden und unermüdlichen Einsatz bei der Fertigstellung dieses Buches. Alexandra und Inken von O'Reilly danken wir für ihre Geduld und dafür, dass sie die den Glauben an das Buch bis zuletzt behalten haben.

KAPITEL 1

Der erste Eindruck

Ein paar ruhige Minuten sind gefunden, Ablenkungsweltmeister wie die Twitters und Facebooks dieser Welt wurden zum Schweigen gebracht, vor Ihnen befindet sich neben der eingangs im Detail beschriebenen Magento-Entwicklungsumgebung nur noch dieses Buch und wahlweise eine Tasse Ihres bevorzugten Heißgetränks. Kurzum, es spricht nichts dagegen, in die Tiefen Magentos vorzudringen.

In diesem Kapitel möchten wir Ihnen einen ersten Eindruck von Magentos Funktionsweise und innerem Aufbau vermitteln. Dabei gehen wir auf das zugrunde liegende Ordnungsprinzip und die Benennungskonventionen genau so ein wie auf seine Verzeichnisstruktur und seine Gliederung in funktionale Einheiten, die sogenannten *Module*. In den nachfolgenden Kapiteln beschäftigen wir uns mit den einzelnen Bestandteilen des Systems im Detail ein und werden diese im zweiten Teil des Buchs anhand praktischer Beispiele weiter vertiefen.

Es gibt grob gesagt zwei verschiedene Wege, sich dem Aufbau von Magento zu nähern und Ihnen damit – wie es auch der Titel dieses Kapitels andeutet – einen ersten Eindruck von diesem System zu vermitteln. Zum einen lassen sich auf Verzeichnis- und Dateiebene wichtige Zusammenhänge verdeutlichen, ganz nach dem Motto: Wo finde ich was? Alternativ kann man sich dem System auch über das funktionale Prinzip des sogenannten *MVC-Patterns* nähern und zeigen, welche Teile des Systems für die Ausgabe, welche für die logische Steuerung und welche für die Datenhaltung verantwortlich sind. Wir haben uns dazu entschlossen, in dieser Einführung beide Wege miteinander zu kombinieren, um Ihnen den bestmöglichen Zugang zu Magento bieten zu können.

Dazu werden wir als Erstes das zugrunde liegende Zend Framework kurz umreißen, da viele Ordnungsprinzipien und Codestandards von Magento stark an dieses Framework angelehnt sind. Danach beleuchten wir den internen Aufbau eines Magento-Moduls und wie sich dieser in die Gesamtstruktur der Magento-Applikation einfügt. In diesem Zusammenhang werden Sie auch erkennen, wie sich darin die Abstammung vom Zend

Framework widerspiegelt. In der Diskussion der Magento-Verzeichnisstruktur gehen wir auf diejenigen Verzeichnisse ein, die für die tägliche Arbeit mit Magento relevant sind.

Anschließend werfen wir einen Blick auf das sogenannten *Model-View-Controller-(MVC-) Pattern* und wie es in Magento umgesetzt wird. Anhand dieses Patterns werden wir die Komponenten der Magento-Systemlogik erläutern. Mit dieser Basis werden Sie in der Lage sein, den Requestzyklus im Abschnitt »Requestzyklus«, auf Seite 15 nachzuvollziehen, der alle beschriebenen Elemente enthält und diese zueinander in Beziehung setzt.

Das Ziel dieses Kapitels ist es demnach, dass Sie die wichtigsten Begrifflichkeiten aus dem Magento-Universum kennengelernt haben und anhand des Requestzyklus wissen, wie sie einzuordnen sind. Dies bezieht sich sowohl auf die Platzierung innerhalb des Programmablaufs als auch auf Magentos Datei- und Verzeichnisstruktur.

Das Zend Framework

Magentos zugrunde liegende Struktur ist weder vom Himmel gefallen, noch stellt sie per se eine Revolution in der Programmiertechnik dar. Vielmehr orientiert sie sich an etablierten Industriestandards und *Best Practices* moderner, objektorientierter PHP-Programmierung. Sie leitet sich hauptsächlich vom Zend Framework (ZF) ab, das zu den am häufigsten genutzten PHP-Frameworks zählt.

Wozu sollte man überhaupt ein Framework benutzen, und aus welchem Grund entschloss sich Varien vor einigen Jahren, ihre Shopsoftware Magento auf einem solchen basieren zu lassen? Die Antwort auf beide Fragen ist im Grunde dieselbe: Weil es die logische Konsequenz aus den Erfahrungen moderner Webentwicklung ist. Mittlerweile befeuern PHP-basierte Applikationen gut besuchte Websites überall auf der Welt und müssen aus diesem Grund hochperformant und sicher sein; außerdem müssen Neuentwicklungen schnell zu erstellen sein, um den Anschluss an den sich immer schneller entwickelnden Markt nicht zu verlieren. Um nicht jedes Mal das Rad neu erfinden zu müssen, besteht daher der nächste Schritt darin, auf ein standardisiertes Framework zurückzugreifen, um wiederkehrende Aufgaben effizient erledigen zu können. Dazu gehören beispielsweise der Versand von E-Mails, diverse Datenbankkonnektoren sowie PDF- und Formularerstellung. Da moderne Webapplikationen längst nicht mehr ohne zumindest eine Prise AJAX auskommen, sind JavaScript-Frameworks wie *Dojo* ebenfalls angebunden.

Die Entwickler von Magento haben also nicht versucht, das Rad komplett zu erfinden, sondern sich dafür entschieden, auf ein bereits existierendes Framework aufzusetzen und von der Stabilität und Flexibilität der vorhandenen Komponenten, der guten Dokumentation und nicht zuletzt der engagierten Community zu profitieren. Falls Sie das eine oder andere Projekt bereits mit dem Zend Framework umgesetzt haben, sind Ihre Voraussetzungen gut, auch in Magento das Gewünschte umzusetzen.

Zur Geschichte des Zend Framework

Wie es der Name schon vermuten lässt, wurde das angesprochene Framework von der Firma Zend erdacht und entwickelt, die gemeinhin als *die* PHP-Firma gilt und mit der Entwicklung von PHP 5 wesentlich zur aktuellen Version der Sprache beigetragen hat sowie dazu, dass PHP auch auf Enterprise-Niveau seinen Außenseiterstatus verloren hat. Das ZF wurde erstmalig im Herbst 2005 vorgestellt und im Frühjahr 2006 als Pre-Alpha-Version 0.1.1 unter der BSD-Lizenz veröffentlicht. Die erste produktive Version 1.0 erschien ein gutes Jahr später. Das Framework basiert auf PHP 5 und ist strikt objektorientiert aufgebaut. Zur Drucklegung dieses Buchs ist die aktuelle Programmversion die 1.10. Die mit Magento 1.4 genutzte Version ist die Version 1.9.6.

Im Zend Framework und damit auch in Magento hat so ziemlich alles seine Ordnung. Beide folgen dem sogenannten Convention over Configuration-(CoC-)Paradigma, d.h., es wird möglichst vermieden, unnötige programmiertechnische Extravaganzen – z.B. bei der Benennung von Klassen und Methoden – einzusetzen, und sich an den gängigen Standards orientiert. Trotzdem gibt es – wie Sie im Laufe dieses Buchs sicherlich erkennen werden – Bereiche, in denen Magento eher konfigurationsbasiert arbeitet.

Was programmiertechnische Freigeister, die sich zum ersten Mal mit einem PHP-Framework beschäftigen, möglicherweise zunächst als Einschränkung empfinden, wird sich sehr schnell als vorteilhaft erweisen, da diese Konventionen Zeit und Arbeit sparen. Werfen wir zunächst einen Blick auf einen Klassennamen: `Mage_Customer_Block_Form_Login`.

Diesem Namen liegt ein System zugrunde, das, wenn Sie sich ein wenig damit auseinandergesetzt haben, Ihnen die Arbeit mit Magento sehr erleichtern wird. Die durch Unterstriche voneinander getrennten und jeweils mit einem Großbuchstaben beginnenden Bestandteile des Klassennamens lassen sich wie folgt auflösen:

Mage
: Hierbei handelt es sich um den Namespace, dem das jeweilige Modul zugeordnet ist. Mehr zum Thema Namespaces können Sie in »Code-Pools und Namespaces«, auf Seite 8 nachlesen.

Customer
: Der zweite Teil des Klassennamens bezieht sich auf den Namen des Moduls, zu der die Klasse gehört. In Kapitel 2 erfahren Sie weitere Details zum Aufbau von Magento-Modulen.

Block
: Bei der vorliegenden Klasse handelt sich sich um einen Block. Sie ist daher auch im Blockverzeichnis des jeweiligen Moduls abgelegt. Details zum Thema Blöcke finden Sie in Kapitel 4.

Form
: Dieser Eintrag bedeutet lediglich, dass innerhalb des Blockverzeichnisses noch ein Formverzeichnis existiert, und ist damit ein weiterer Hinweis auf den Speicherort der erwähnten Klasse.

Login
: Dieser letzte Teil bezeichnet den Dateinamen der Klasse.

Dies bedeutet, dass der Klassename – so lang und sperrig er auf den ersten Blick auch wirkt – gleichzeitig seinen eigenen Speicherort enthält, in diesem Fall */app/code/<codepool>/Mage/Customer/Block/Form/Login.php*.

Wir werden uns in diesem Buch lediglich auf die jeweiligen Namen der Klassen beziehen; da der Speicherort der entsprechenen Verzeichnisse daraus resultiert, sind Sie damit in der Lage, die jeweiligen Dateien zu finden und die Erläuterungen nachzuvollziehen.

Konventionen und Best Practices

Entwickler, die sich schon eingehender mit dem ZF beschäftigt haben, werden bei der ersten Durchsicht der Magento-Ordnerstruktur und des Quellcodes viele alte Bekannte wiedertreffen. Die Aufteilung der Module in Models, Controller und Helper beispielsweise begegnen einem auch in einer Zend-Applikation; Gleiches gilt für Klassennamen bzw. deren implizierte Fähigkeit, sich über den Autoloader selbst zu laden, da der Klassenname selbst bereits klarmacht, an welcher Stelle des Filesystems sie gespeichert ist.

Granularer Aufbau durch Module

Ein oft genannter Vorteil und einer der wichtigsten Gründe für den anhaltenden Erfolg von Magento ist sein modularer Aufbau. Wie in Kapitel 2 noch im Detail erläutert wird, werden in Magento bestimmte zueinander gehörende Prozesse zu einem Modul gebündelt. Unter einem Modul versteht man demnach eine funktionale Einheit, die unter dem entsprechenden Namen und innerhalb eines bestimmten Namespace (siehe dazu den Abschnitt »Code-Pools und Namespaces«, auf Seite 8) gespeichert ist und Aufgaben wie die Kundenverwaltung und den Bestellprozess übernimmt. Wichtige Module heißen beispielsweise *Catalog*, *Checkout*, *Sales* oder *Customer*.

Für jedes Modul gelten die gleichen Ordnungsprinzipien hinsichtlich seiner hierarchischen Struktur. Verzeichnisse und die darin enthaltenen Dateien müssen in einer bestimmten Weise benannt und angeordnet sein, um die jeweiligen Funktionalitäten für die Magento-Applikation bereitzustellen. Hierbei gibt es Bestandteile, die obligatorisch sind – ein Modul muss beispielsweise immer eine Konfigurationsdatei *config.xml* enthalten – und andere, die je nach Zweck des Moduls zum Einsatz kommen oder auch nicht.

Man könnte nun erwarten, dass jedes Modul so gekapselt ist, dass in einem dedizierten Modulverzeichnis alle zugehörigen Programmbestandteile abgelegt sind. Dies ist jedoch nicht der Fall, denn jedes Modul besteht aus einem funktionalen und einem gestalterischen Teil. Auch wenn wir hier der näheren Erläuterung in Abschnitt »/app/code/«, auf Seite 8 ein wenig vorgreifen: Ersteren finden Sie in der gesamten Magento-Verzeichnisstrukur in */app/code* und Letzteren in */app/design* wieder.

Der Grund liegt – wie wir bei der Diskussion des MVC-Pattern im Abschnitt »Das MVC-Pattern«, auf Seite 11 noch sehen werden – in der Trennung zwischen Programmlogik und dem Teil, der für die Ausgabe und Formatierung (der sogenannte *View*) zuständig ist.

Module vs. Extensions

Sowohl Varien – die US-amerikanische Firma, die Magento konzipiert hat und dessen Entwicklung aktiv vorantreibt – als auch die gesamte restliche Magento-Welt verfährt mit den technischen Begrifflichkeiten leider nicht so einheitlich, wie man sich das wünschen würde. Es herrscht unter anderem Unsicherheit darüber, worin sich Module und Extension unterscheiden bzw. wann genau der eine oder der andere Begriff verwendet werden soll.

Wir haben uns entschlossen, das Wort *Modul* immer dann zu verwenden, wenn es sich um Core-Module handelt, die nach einer Standardinstallation vorhanden und unter */app/code/core/Mage/* abgelegt sind. Demgegenüber verstehen wir unter einer Extension eine Eigenentwicklung, die ebenfalls Funktionalitäten für Magento bereitstellt, jedoch entweder unter *app/code/community* oder unter *app/code/local* auf Ihrem Server abgelegt ist.

Extensions sind exakt so aufgebaut wie Module, deshalb werden diese beiden Begriffe auch gern synonym verwendet. Erstere sind aber in den meisten Fällen ungleich weniger komplex als die Module, die letztlich den Magento-Kern darstellen.

Die Modul-Programmlogik

Wie Sie Laufe der folgenden Kapitel noch genauer sehen werden, gehören zu den Bestandteilen eines Moduls Klassen für die verschiedensten Einsatzbereiche: Sie sorgen beispielsweise dafür, dass relevante Daten (z.B. Produkt- oder Kundendaten) in Objekten gespeichert und damit der Applikation zur Verfügung gestellt werden. Diese Klassen generieren also die Models bzw. Resource-Models, die Ihnen in Kapitel 3 noch begegnen werden. Eine zweite Art von Klassen generiert die sogenannten Blöcke, also Teile der View-Schicht, die für den Transport der durch die Models bereitgestellten Daten zur Ausgabe im Browser verantwortlich sind. Eine dritte Art von Klassen sind die sogenannten Helper-Klassen, die, wie es der Name vermuten lässt, kleinere und generische Hilfsmethoden bereitstellt, die beispielsweise Übersetzungen übernehmen oder Preise korrekt formatieren.

In Abbildung 1-1 erkennen Sie, wie der Programmteil eines typischen Magento-Moduls aufgebaut ist.

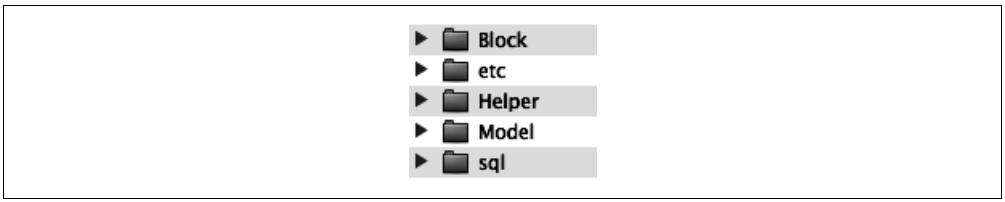


Abbildung 1-1: Verzeichnisstruktur des Programmteils eines Moduls

Neben diesen Klassen finden Sie innerhalb eines Moduls auch eine Reihe von Konfigurationsskripten, die als XML-Dateien die Funktion der einzelnen Module steuern. Last, but not least liegen dort auch Installations- und Update-Skripte, die für die Vorbereitung und Füllung der für das Modul wichtigen Datenbanktabellen zuständig sind.

Im Abschnitt »/app/design/«, auf Seite 9 werden Sie erfahren, in welcher Weise die programmlogischen Bestandteile der Magento-Module ihren Platz im */app/design*-Verzeichnis der Magento-Applikation finden.

Die Modul-Gestaltung

Analog zur Programmlogik jedes Moduls ist auch der Gestaltungsteil in einer fest definierten Weise aufgebaut. Dieser beinhaltet jeweils eine Layoutdatei, mit der die Ausgaben des jeweiligen Moduls vorformatiert bzw. die Strukturblöcke den Inhaltsblöcken zugeordnet werden (siehe Kapitel 4). Ebenso existiert ein Template-Verzeichnis mit einer Reihe von Template-Dateien, über die die HTML-Ausgabe gesteuert wird. Die gestalterischen Teile der Module sind in Magento unter */app/design* abgelegt, worauf wir im weiteren Verlauf dieses Kapitels noch genauer eingehen werden.

Zusammenfassend lässt sich also sagen, dass jedes Magento-Modul seine eigene interne Verzeichnisstruktur hat, die sich in die Gesamtstruktur der Magento-Installation einfügt. Die Struktur ist dabei in einen programmlogischen und einen gestalterischen Teil aufgebrochen, um der Trennung zwischen Logik und Gestaltung Rechnung zu tragen. Im folgenden Abschnitt werden wir nun die Perspektive ändern und Ihnen näherbringen, wie sich die Bestandteile der einzelnen Module in die gesamte Verzeichnisstruktur der Magento-Applikation einfügen.

Die Verzeichnisstruktur von Magento

Frisch nach der Installation werden Sie eine Verzeichnisstruktur auf Ihrem System sehen, in der es insgesamt 11 Verzeichnisse und 13 Dateien auf der Root-Ebene gibt. Was auf den ersten Blick wie eine überschaubare – ja fast bescheidene – Sammlung von Programmdateien aussieht, entpuppt sich nach wenigen Klicks als sehr komplexes Gebilde,

das jedoch einem strengen und gut nachvollziehbaren Ordnungsprinzip folgt. Hier hat jede Klasse und jede Konfigurationsdatei ihren fest vorgegebenen Speicherort, um das Arbeiten und das spätere Auffinden zu erleichtern (siehe Abbildung 1-2).

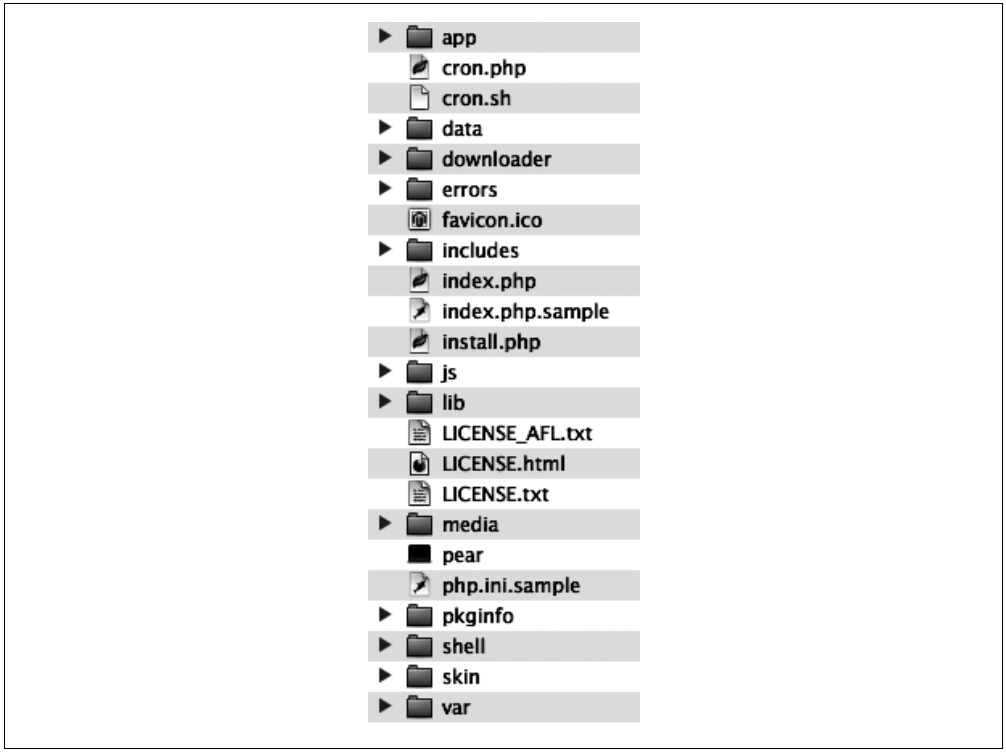


Abbildung 1-2: Verzeichnisstruktur einer neuen Magento-Installation

Insbesondere der */app*-Ordner hat es in sich: Hier sind die Module von Magento untergebracht, die in ihrer Gesamtheit die Funktionalität der E-Commerce-Software ausmachen. Jedes dieser Module besteht seinerseits aus zahlreichen Unterverzeichnissen mit allen Models, Blöcken, Layouts und Templates, aus denen sich Magento zusammensetzt. Mehr zu diesen einzelnen Bausteinen erfahren Sie in den nachfolgenden Abschnitten, weitere Details finden Sie in den nächsten Kapiteln.

Wir werden nicht auf jedes Verzeichnis und auf jedes einzelne Skript einer Magento-Installation eingehen – schließlich ist der Zweck dieses Buchs, Sie in einem vertretbaren Zeitraum mit den wichtigsten Elementen der Magento-Entwicklung vertraut zu machen, und nicht, in einer mehrbändigen Sammlung von Fachliteratur zu münden.

Wir beginnen damit, uns die Verzeichnisse */app/code* und *app/design* genauer anzusehen, da Sie sich während der Magento-Entwicklung hauptsächlich mit den darin gespeicherten Elementen beschäftigen werden.

/app/code/

In diesem Verzeichnis finden Sie sozusagen den Motor, der die Magento-Maschinerie antreibt. Ein großer Teil der Funktionalität, die in ihrer Gesamtheit letztlich den Onlineshop generiert, ist in diesem Verzeichnis gespeichert.

Code-Pools und Namespaces

Unter einem *Code-Pool* versteht man im Magento-Universum vereinfacht gesagt ein Verzeichnis, in dem Programmteile gespeichert werden. Bildlich gesprochen, handelt es sich dabei um drei verschiedene Behälter namens *community*, *core* und *local*, die Namespaces enthalten, die wiederum mit Modulen gefüllt sind. Während der Ausführung arbeitet Magento nacheinander jeden der drei Behälter ab, um ein Modul zu laden und es in seinen Programmablauf zu integrieren. Wie Sie im weiteren Verlauf noch sehen werden, hat es theoretisch keine Bewandtnis, in welchem Pool Sie Ihre neuen Programmierungen anlegen, praktisch jedoch hat es für die Stabilität und Updatefähigkeit sehr wohl eine wichtige Bedeutung.

Die gezeigten Codebehälter unterscheiden sich also strukturell und inhaltlich nicht voneinander, vielmehr ist die Reihenfolge wichtig, in der die Behälter abgearbeitet werden. Genaueres zu dieser Abarbeitungsreihenfolge finden Sie im Abschnitt »Der PHP include_path«, auf Seite 58.

Im Zusammenhang mit den Code-Pools sei ein weiteres Ordnungskriterium genannt, das der sogenannten *Namespaces*. Magento ist so aufgebaut, dass Sie individuelle programmierte Extensions in einem Verzeichnis unterbringen können, das einen beliebigen Namen trägt. Das bietet Ihnen die Möglichkeit, zu einem Projekt gehörende Extensions zu kapseln und sie leichter auffindbar zu machen. Wenn Sie Extensions via *MagentoConnect* herunterladen, werden Sie feststellen, dass viele Extension-Anbieter ebenfalls von dieser sinnvollen Möglichkeit Gebrauch machen und ihre Extension in ein Verzeichnis, das den Firmennamen trägt, ablegen. Core-Module erkennen Sie in diesem Zusammenhang daran, dass sie im *Mage*-Namespace abgelegt sind

Andere Namespaces sind Ihnen bei Ihrer bisherigen Arbeit wahrscheinlich begegnet, als Sie eine neue Extension über *MagentoConnect* installiert haben. Dabei handelt es sich um die einfache Möglichkeit, Ihrer eigenen Shopinstallation über Magentos hauseigenen Extension-Marktplatz per Mausklick neue Funktionalitäten zu kredenzen. Alles was Sie in dem Zusammenhang tun müssen, ist, über den *MagentoConnect*-Manager in Ihrer Systemkonfiguration den entsprechenden Extension-Code einzutragen und den Installationsprozess zu starten. Nach dieser Installation werden die hinzugekommenen

Extensions in den entsprechenden Code-Pools und den jeweiligen Namespaces angezeigt.

Spätestens jetzt werden Sie erkannt haben, dass hier auch die programmlogischen Bestandteile der einzelnen Module abgelegt sind. Jedes Modul – genauer gesagt der Teil, der für dessen Funktionalität verantwortlich ist – wird also unter seinem eigenen Namen im vorher festgelegten Namespace und Code-Pool gespeichert.

Wie schon erwähnt, werden in Magento die Bestandteile eines Moduls, die die Funktionalität bereitstellen, von denen getrennt, die für die Präsentation zuständig sind. Im nächsten Abschnitt erfahren Sie, an welcher Stelle der zweite, gestalterische Teil eines Moduls innerhalb der gesamten Verzeichnisstruktur abgelegt wird.

/app/design/

Analog zum */app/code/*-Verzeichnis werden Ihnen im */app/design/*-Verzeichnis die Teile der Module begegnen, die verantwortlich sind für Browserausgaben – was nutzt die schönste interne Datenstruktur, wenn sie nicht für den Besucher attraktiv und übersichtlich aufbereitet wird. Mit dem Öffnen von */app/design/* befinden Sie sich damit gleichzeitig mitten in Magentos Theme- und Template-System (siehe Abbildung 1-3).

Abbildung 1-3: Der Aufbau von /app/design/

Packages (Interfaces) und Themes

Analog zu den Code-Pools und Namespaces des */app/code*-Verzeichnisses finden sich auch in */app/design* weitere Hierarchieebenen. So unterscheidet Magento zunächst, für welchen Ausgabenbereich die Gestaltungsdateien gelten sollen. Es gibt drei Möglichkeiten:

adminhtml
: In diesem Teil des Designverzeichnisses werden Gestaltungsänderungen gespeichert, die das Admin-Panel betreffen. Möchten Sie beispielsweise den Aufbau der verschiedenen Eingabemöglicheiten in der Shopverwaltung optisch anpassen, ist dieses Verzeichnis für Sie interessant.

frontend
: Hier finden Sie sämtliche Gestaltungsdateien, die zusammen genommen das komplette Design Ihres Onlineshops steuern. Im Folgenden werden wir uns hauptsächlich auf dieses Verzeichnis konzentrieren, wenn es um die Anpassung des Shopdesigns geht.

install
: Wie der Name schon vermuten lässt, ist dieser Bereich dazu gedacht, etwaige Gestaltungsänderungen des Installationsprozesses abzubilden.

Innerhalb des */frontend*-Verzeichnisses befindet sich nach der Installation ein Unterordner namens */default*. In der Magento-Terminologie handelt es sich dabei um ein *Package* (oder auch *Interface*), das mehrere *Themes* in sich vereint. Es ist also eine übergeordnete Struktur, die – ähnlich den Code-Pools – dazu genutzt werden kann, mehrere kleinere Einheiten zusammenzufassen. Erfahrungsgemäß werden jedoch auch bei größeren E-Commerce-Projekten selten Situationen auftreten, in denen das Anlegen eines neuen Interface nötig ist. In den meisten Fällen ist es vollkommen ausreichend, mehrere verschiedene Themes zu erstellen, von denen es in einer Standardinstallation ebenfalls eins gibt. Sie ahnen es bereits, es heißt natürlich auch *default*.

Für die weiteren Betrachtungen ist also das Verzeichnis */app/design/frontend/base/default/* interessant. Im nächsten Abschnitt lernen Sie den Inhalt dieses Theme-Verzeichnisses genauer kennen.

Layouts und Templates

Wie so oft, wenn es in Magento um Konfigurationen geht, kommt XML zum Einsatz, und so verwundert es auch nicht, dass der grobe Aufbau des späteren Magento-Shops mittels XML-Dateien gesteuert wird. Jedes Modul hat seine eigene Layoutdatei, die dessen Namen trägt und unter */app/design/frontend/base/default/layout/catalog.xml* abgelegt ist. In diesem Fall wird das Modul *Catalog* hinsichtlich seiner Browserausgaben vorstrukturiert. Wie Sie in Kapitel 4 sehen werden, sind die Grundlage für die Gestaltung der Module die sogenannten Blöcke, von denen es in Magento vor allem zwei Arten gibt: Strukturblöcke und Inhaltsblöcke. Unter einem Strukturblock versteht man einen festen Bereich der Seitengestaltung wie beispielsweise den Kopf- oder Fußbereich oder die rechte bzw. linke Seitenleiste. Innerhalb dieser Struktur werden die Ausgaben der Module als Inhaltsblöcke angeordnet. So wird beispielsweise im Checkout-Modul ein Inhaltsblock erzeugt, der eine verkleinerte Warenkorbvorschau darstellt. Diese Vorschau wird anschließend beispielsweise dem Strukturblock *Seitenleiste links* zugeordnet.

Diese Zuordnung wird über Layoutdateien im XML-Format gesteuert. Sie definieren darüber hinaus, mit welchen Template-Dateien die Ausgabe der Inhaltsblöcke formatiert wird. Für jedes Modul findet man daher auch ein nach dem Modul benanntes Template-Verzeichnis, in dem die Template-Dateien abgelegt sind; das *Catalog*-Modul unter */app/code/core/Catalog/* findet demnach seine Entsprechung im Template-Verzeichnis */app/design/frontend/base/default/template/catalog/*.