Magento

Erfolgreich mit dem ersten Online-Shop

SUSANNE ANGELI WOLFGANG KUNDLER

Markt+Technik

CD

Shop-Software Magento

KAPITEL 3
Shop-Design und SEO

Im aktuellen Kapitel vermitteln wir Ihnen das Basiswissen für die Arbeit mit einem **Layout**. Anhand einiger Praxisbeispiele und Anregungen finden Sie den schnellen Einstieg zu einem individuellen Design. Im weiteren Verlauf des Kapitels bekommen Sie dann einen ersten Einblick, wie Sie das begonnene Layout mit einfachen Mitteln suchmaschinenfreundlicher gestalten. Erst wenn beides perfekt harmoniert, schaffen Sie eine gute Basis, um erfolgreich und profitabel im Online-Handel zu agieren.

Ihr Ziel muss es sein, etwa 50 bis 80% des Traffics über Suchmaschinen zu generieren. Je größer eine Website, desto niedriger wird das Seitenzugriffsverhältnis zwischen SEO einerseits sowie SEM und Direktzugriffen andererseits sein. Dies gelingt jedoch nur dann, wenn Sie das Design SEO-technisch verbessern. Steht das Layout, dann konzentrieren Sie sich gezielt auf Vertrieb und Vermarktung Ihres Shops. Mit Suchmaschinen-Marketing kommen zusätzliche Besucher auf Ihre Seiten. Das kann aber nur dann rentabel funktionieren, wenn die Marge stimmt. Mehr zum Thema **Marketing** lesen Sie in *Kapitel 5*.

3.1 Leitfaden zum eigenen Shop-Design

SEO-optimierter Magento-Shop

Wer kostengünstig den ersten eigenen Shop aufbauen will, der muss sich unbedingt mit dem Thema Layout befassen. Sicherlich kann Ihnen ein externer Programmierer oder Webdesigner bei der ersten Umsetzung behilflich sein. Doch einen richtig erfolgreichen Online-Shop machen oftmals unzählige kleine Änderungen aus. Zwar ist *Magento* extrem fortschrittlich für SEO vorbereitet, dennoch gibt es einiges zu tun. Natürlich ist nicht jeder guter Webdesigner gleichzeitig ein perfekter SEO-Profi. Daher werden Sie früher oder später selbst die vielen kleinen Layout- und SEO-Verbesserungen erledigen. Oftmals geht es nur um Textauszeichnungen (Überschriften, Fettdruck oder alt-/title-Tags) oder die Textanzeige (Hyperlinks, Textänderungen oder Positionierung). Viele Dinge lassen sich über das Backend administrieren, doch leider nicht alle. Aber genau diese unscheinbaren Kleinigkeiten machen den Unterschied zwischen Mittelmaß und Erfolg aus.

Begriffsdefinitionen zum Webdesign

Eignen Sie sich daher im Laufe der Zeit die wichtigsten Layouttechniken an und sparen Sie Geld und Zeit. Denn alles, was Sie selbst erledigen, kostet weniger und geht meistens schneller. Vielleicht möchten Sie die Optik der Breadcrumb ändern von »Home / Electronics / Cameras« in »Home ? Electronics ? Cameras«. Die größte Hürde dabei ist eigentlich herausfinden, an welcher Stelle Sie das ändern müssen. Auf den kommenden Seiten gehen wir zu Anfang auf Begriffsdefinitionen rund um das Shop-Design ein, wir zeigen Hilfsmittel und beschreiben Ihnen, wie Sie ein eigenes Design erstellen. Es ist kein Zufall, dass wir in diesem Kapitel die beiden Themen **Webdesign** und Suchmaschinenoptimierung kombinieren. Die Kunst ist es, den folgenden Spagat zu schaffen:

- Design: Besucher anlocken und in kaufende Kunden verwandeln
- SEO: *Google* mit relevanten Keywords und passenden Inhalten füttern

Design, SEO und Usability

Konzentrieren Sie sich zu sehr auf das optische **Design** und technische Layout-Finessen, entsteht zwar ein optisch ansprechender Shop, Sie vernachlässigen aber gleichzeitig die SEO-Grundregeln. Wer stattdessen die **Suchmaschinenoptimierung** zu sehr in den Mittelpunkt stellt, der erstellt womöglich einen wenig usability-freundlichen Shop, der keine Kunden anlockt. Das Tolle an *Magento* ist: Es bietet Ihnen ausgefeilte und durchdachte Techniken sowohl für das Design als auch für die Suchmaschinenoptimierung des Shops.

www. . . . . .

magentocommerce.com/design_guide
MagentoCommerce (Designers Guide to Magento)

magentocommerce.com/media/screencasts/designers-guide-1/view
MagentoCommerce (**Screencast**: Designers Guide – Creating Magento Themes)

magentocommerce.com/wiki/how-to/designing/designing-for-magento
MagentoCommerce Wiki (Designing for Magento)

magentocommerce.com/support/magento-user-guide-book
MagentoCommerce (Offizieller Magento User Guide [engl.], kostet 20 US$)

Info . . . . . .

Freundlicherweise stand uns in diesem Kapitel der in Potsdam wohnende Daniel Sasse (alias ds_1984) mit Rat und Tat zur Seite. Er befasst sich schon seit Ende 2008 mit Magento und hat bereits mehrere Shops erfolgreich geplant und umgesetzt. Aktuell arbeitet er in einem mittelständischen Unternehmen im Bereich Webentwicklung und Marketing. Sehr fleißig engagiert er sich im deutschsprachigen MagentoCommerce-Forum und als Autor für das Praxisblog `mxperts.de`.

Laut eigener Aussage trifft er im Forum auf Menschen mit völlig unterschiedlichen Vorkenntnissen, Meinungen und teilweise sehr spannenden Vorhaben. Doch eine Sache vereint alle im Forum – sie wollen ihre Aufgaben mit Magento lösen und das führt zu einem harmonischen Miteinander und auch einer sehr hohen Hilfsbereitschaft.

Begriffe des Shop-Designs

Deutsch/ Englisch Terminologie

Zum besseren Verständnis der Thematik ist es wichtig, dass Sie die einzelnen Begriffe kennenlernen. Eine einheitlich genutzte Terminologie ist besonders hilfreich, wenn Sie im deutsch- oder englischsprachigen Forum entweder selber Fragen stellen oder nach Problemlösungen suchen. Im folgenden Abschnitt lernen Sie diese Begrifflichkeiten kennen und unterscheiden:

- Website, Store und Store View
- Interfaces auf Website- oder Store View-Ebene
- Standard-/Non-default-Thema: Layout, Locale, Template und Skin
- Strukturelle Blöcke (Structural Blocks) und Content-Blöcke (Content Blocks)

Website, Store und Store View

Der Begriff **Website** definiert sich im Rahmen von *Magento* etwas anders, als Sie es bisher gewohnt sind. Im Allgemeinen ist es eher üblich, dass der Inhalt aller Einzelseiten auf einer Domain als Website betrachtet wird. Das Shop-System versteht den Begriff hingegen etwas weitläufiger. Eine Website ist hier eine Sammlung mehrerer Stores, die sich die gleichen Kunden und Bestellinformationen teilen. Ein **Store** wiederum ist eine Sammlung einzelner **Store Views**. Nähere Details zum **Multistore**-Konzept finden Sie in *Kapitel 1.2*.

Unterschiedliche Kundensegmente bedienen

Mit Store Views lassen sich nicht nur verschiedene Sprachen realisieren, sondern auch unterschiedliche Designs anbieten, beispielsweise für Frauen, Männer und Kinder. Zudem eignen sie sich ebenso, um verschiedene **Kundensegmente** zu bedienen, wie Endkunden und registrierte Reseller. Allerdings benötigen Sie dazu zwei getrennte Websites, denn innerhalb einer Website ist der Preis eines Produkts in allen Store Views identisch. Natürlich müssen Sie die Produkte gleichzeitig mehreren Websites zuordnen. Nur dann lohnt sich der Wechsel des **Katalogpreis-Gültigkeitsbereichs** (Catalog Price Scope) von »Global« auf »Website«. Die Änderung nehmen Sie vor unter »System › Konfiguration › Katalog › Preis« (System › Konfiguration › Catalog › Prices). Rufen Sie danach einen beliebigen Artikel in der Produktverwaltung auf. Nachdem Sie bei Produktinformationen die Preise öffnen, wählen Sie links oben im Dropdown-Menü eine Store View aus. Entfernen Sie den Haken vor »Verwende Standardwert« (Use Default Value). Nun ist es möglich, in der Store View einen anderen Preis festzulegen.

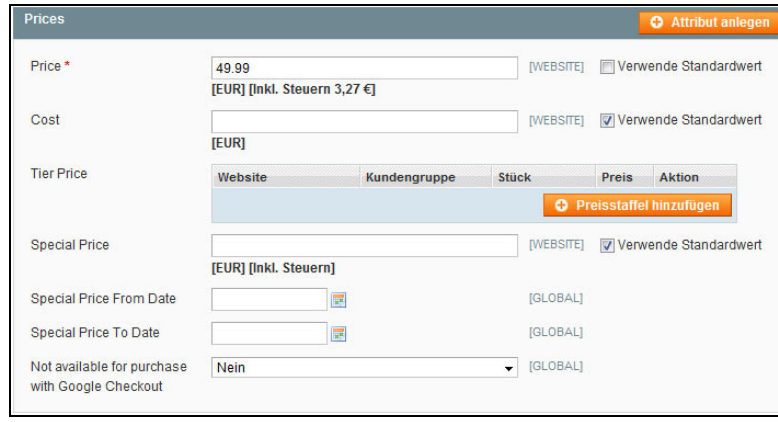

Abbildung 3.1: Website-Preis ändern auf Store View-Ebene

Interfaces auf Website- oder Store View-Ebene

Sammlung einzelner Themen

Ein einzelnes **Interface** ist eine Sammlung von Themen. Die unterschiedlichen Themen bestimmen das optische Aussehen und die Frontend-Funktionalitäten Ihrer Stores. Die Zuweisung eines Interface erfolgt entweder auf Website-Ebene und/oder auf Store View-Ebene. Diese Art der Gestaltung unterscheidet sich folgendermaßen:

- **Website-Ebene**: Alle Stores erben das Interface der übergeordneten Website.
- Diese Einstellung ist die richtige Wahl, wenn Sie ein einheitliches Layout (Corporate Design) für Ihren kompletten Shop wünschen.
- **Store View-Ebene**: Jede Store View erhält ihr eigenes Interface.
- Bei dieser Variante haben Sie die Möglichkeit, jedem einzelnen Store ein individuelles Aussehen zuzuordnen.

Design einem Store zuteilen

Haben Sie einem Store bereits ein Design zugeteilt, so wird logischerweise diese Optik gegenüber der übergeordneten Website bevorzugt. Anhand zweier optischer Beispiele möchten wir Ihnen die Arbeitsweise besser verdeutlichen. Wenn Sie die Abbildung 3.2 genauer betrachten, sehen Sie am besten, wo die Unterschiede bei der Zuordnung des Interface auf Website- und Store View-Ebene liegen.

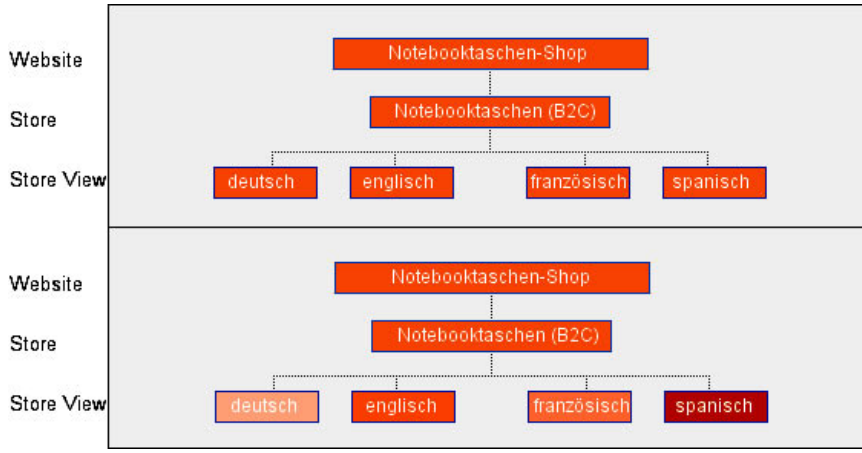

Abbildung 3.2: Designzuweisung auf Website- und Store View-Ebene

Sobald Sie ein eigenes Design umsetzen möchten, erstellen Sie als Erstes ein eigenes Interface (z.B. myInterface) und ein eigenes Thema (z.B. myTheme). Kopieren Sie aber unbedingt auch das default-Theme in den neu angelegten Interface-Ordner, sonst entstehen durchaus ungewöhnliche Fehler. Extrem wichtig ist es, die Umbenennung oder Umstrukturierung nicht nur im Ordner `/app`, sondern ebenso im Ordner `/skin` vorzunehmen.

Interface im Backend ändern

Die Zuweisung eines Interface nehmen Sie im Administrationsbereich vor, unter »System › Konfiguration › Gestaltung« (System › Configuration › Design). Dort öffnen Sie »**Paket**« (**Package**) und ändern den Interface-Namen im Textfeld »Aktueller Paketname« (Current package name). Der standardmäßig voreingestellte Interface-Name lautet, genauso wie das erste Thema, **default**. Zu Beginn steht der »Aktuelle Konfigurationsbereich« (Current Configuration Scope) auf **Standardkonfiguration** (Default Config), demzufolge auf Website-Ebene.

Step . . . . . .

1. Ändern Sie den aktuellen Konfigurationsbereich auf eine Store View.
2. Tragen Sie den Interface-Namen ein, in dem Ihr Thema liegt.
3. Notieren Sie den Namen Ihres Themas im Standardtextfeld.

Links oben finden Sie den vorkonfigurierten aktuellen Konfigurationsbereich. Am schnellsten geht es, wenn Sie dort die Website auswählen und bei Bedarf gleichzeitig für alle darin liegenden Store Views den Interface-Namen abändern. Unter »Paket« tragen Sie den neuen Namen des Interface ein, in dem Sie alle Themen hinterlegt haben, die Sie für diese Website benötigen. Diese Änderung ist nur möglich, wenn Sie eine beliebige Store View auswählen und den Haken vor »Website verwenden« (Use website) entfernen. Erscheint die **Fehlermeldung** »*package with this name does not exist and cannot be set*«, dann sollten Sie prüfen, ob im Verzeichnis `/app/design/frontend/` der Interface-Name überhaupt existiert oder vielleicht falsch geschrieben ist.

Hierarchie und Rang von Layouts

Im nächsten Schritt ordnen Sie der jeweiligen Store View eines der Themen aus Ihrem Interface zu. Suchen Sie also jetzt beim Konfigurationsbereich nach der gewünschten Store View, die ein neues Design bekommen soll. Bleibt das Feld »Standard« (Default) leer, kommt automatisch das Thema »default« zum Einsatz. Sobald Sie den Namen eines anderen Themas eintragen, erhält dieses in der **Hierarchie** einen höheren **Rang** und die darin enthaltenen Layoutanpassungen werden bevorzugt.

Daher brauchen Sie nicht alle Daten zu duplizieren, wenn Sie ein eigenes Thema erstellen. Es reicht vollkommen aus, nur geänderte Dateien im eigenen Thema abzulegen. Fehlt ein bestimmter Dateityp, egal ob CSS, JavaScript, Datei oder Bild, verwendet das Shop-System automatisch den Inhalt des »default«-Themas. Der Shop bekommt das neue Design, sobald Sie die geänderte Konfiguration abspeichern und das Shop-Frontend neu laden.

Standard-Thema vs. Non-default-Thema

Layout-, Template-, Locale- und Skin-Datei

Ein **Thema** (Theme) ist eine Kombination aus Layout-, Template-, Locale- und/oder Skin-Dateien, die das visuelle Aussehen Ihres Shops beeinflussen. Ein komplettes Thema besteht meist aus Dateien der beiden Ordner `/app` (layout, locale und template) und `/skin` (css, images und js). Natürlich bekommt ein Thema einen einheitlichen Ordnernamen im zugehörigen Interface zugeteilt. *Magento* bietet Ihnen die Möglichkeit, gleichzeitig verschiedene Themes zu laden, wobei das Shop-System zwei Typen unterscheidet:

- **Standard-Thema** (Default theme): Jedes Interface bindet standardmäßig ein Thema namens »default« ein, welches das Hauptthema (Main Theme) darstellt.
- **Non-default-Thema**: Dies ist ein nicht standardmäßig eingebundenes Theme, welches nur so viele Theme-Dateien wie nötig beinhaltet. Sie brauchen keine komplette Kopie des Default Theme. Es ist völlig ausreichend, nur die Dateien in einem solchen Thema abzuspeichern, die Sie tatsächlich ändern möchten.

Das Standard-Thema beinhaltet bereits alle erforderlichen Dateien, um einen Shop-Store fehlerfrei zu betreiben. Es nimmt in der **Themenhierarchie** allerdings die niedrigste Stufe ein. Ein Non-default-Thema bekommt eine höhere Stufe zugeteilt. Es ist möglich, aber weniger empfehlenswert, direkt die Dateien im Standard-Thema zu bearbeiten. Die bessere Alternative ist, zusätzlich einige ausgewählte Dateien im Non-default-Thema abzulegen und diese zu bearbeiten. Gemäß der höheren Hierarchie überschreibt eine identisch benannte Datei die andere Datei im Standard-Thema.

Ein eigenes Thema mit dem Namen »mytheme« besteht aus den nachfolgenden Inhalten. Die einzelnen Dateien finden Sie im Ordner `/app/design/frontend/`:

>> **Layout:** `/myinterface/mytheme/layout/`

Darin finden Sie die grundlegenden XML-Dateien. Diese beinhalten die Blockstruktur einiger Seiten, sie beeinflussen Meta-Daten und Seitencodierung.

>> **Locale:** `/myinterface/mytheme/locale/`

Dort befindet sich pro Sprache eine simple Textdatei (translate.csv), die Übersetzungen als Store-Kopie enthält.

>> **Template:** `/myinterface/mytheme/template/`

Hier liegen die *.phtml-Dateien mit den Textauszeichnungen und anderen wichtigen PHP-Codezeilen, die die Logik für das optische Aussehen erzeugen.

Im **Skin**-Ordner `/skin/frontend/myinterface/mytheme/` liegen noch ein paar blockspezifische JavaScript-, CSS- und Bilddateien, die der Shop für das Thema benötigt.

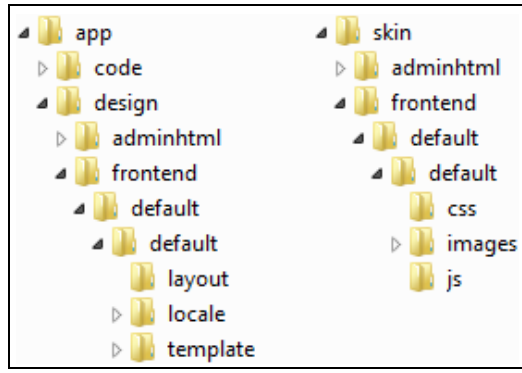

Abbildung 3.3: /app- und /skin-Ordner des default-Theme

Wozu strukturelle Blöcke und Content-Blöcke?

Funktionelles und visuelles Hilfsmittel

Blöcke sind eine wesentliche Möglichkeit, verschiedene Funktionen eines *Magento*-Shops darzustellen. Dadurch erhalten Sie die Gelegenheit, modulartig die gewünschten Informationen zusammenzustellen.

Die Blöcke dienen dabei gleichzeitig als visuelles und funktionelles Hilfsmittel. Das Shop-System unterscheidet zwei Arten von Blöcken, die Ihnen beide gleichermaßen helfen, die visuelle Ausgabe zu erstellen:

- **Strukturelle Blöcke** (Structural Blocks): dienen lediglich zur visuellen Strukturierung eines Store, also das Layout wie: Header, Main Content, Footer, linke oder rechte Spalte
- **Content-Blöcke** (Content Blocks): erzeugen den tatsächlichen Inhalt innerhalb eines strukturellen Blocks. Die einzelnen Template-Dateien erstellen den (X)HTML-Quellcode, der im übergeordneten strukturellen Block eingefügt wird. Eigene Content-Blöcke sind beispielsweise Kategorieliste, Produkt-Tags oder Produkt-Listing.

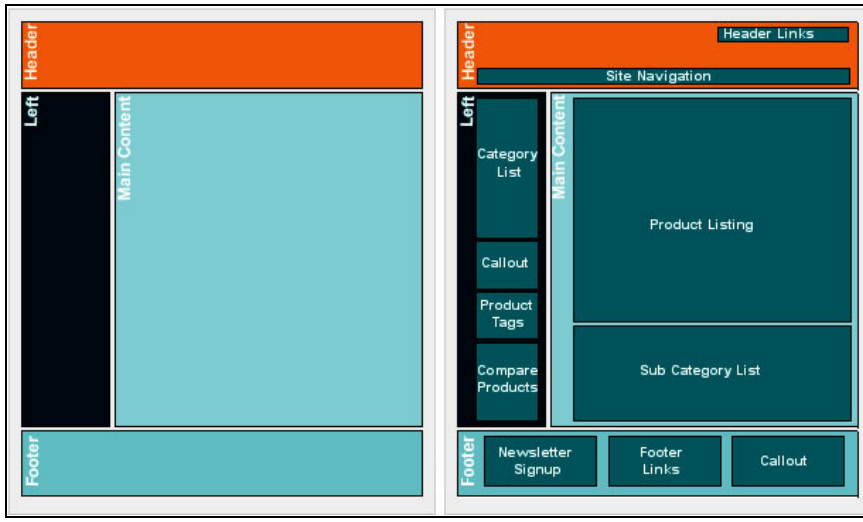

Abbildung 3.4: Strukturelle Blöcke im Vergleich zu Content-Blöcken

Zweispaltiges Aussehen der Seite

Das **Default-Layout** finden Sie im `/app/design/frontend/`-Verzeichnis der Datei `/layout/page.xml`, damit laden Sie bereits die häufigsten Seiteninhalte. Wie in vielen Dateien sind auch hier die Inhalte in Form von XML-Daten strukturiert. Im Ordner `/template/page/` befinden sich einige strukturelle Blockelemente. Der Quellcode-Auszug aus der Datei `2columns-right.phtml` demonstriert Ihnen in Listing 3.1 die Struktur einer zweispaltigen Seite, ähnlich wie in Abbildung 3.4. Darin sehen Sie deutlich, wie und wo die Layoutelemente angeordnet werden. Strukturelle Blöcke bestehen aus (X)HTML und PHP, daher erhalten diese die Dateiendung *.phtml.

Listing 3.1: Strukturelle Blöcke von 2columns-right.phtml

```
<!-- start header -->
<div class="header">
  <?php echo $this->getChildHtml('header') ?>
</div>
<!-- end header -->
<!-- start middle -->
<div class="middle-container">
  <div class="middle col-2-right-layout">
    <?php echo $this->getChildHtml('breadcrumbs') ?>
    <!-- start center -->
    <div id="main" class="col-main">
      <?php echo $this->getChildHtml('global_messages') ?>
      <?php echo $this->getChildHtml('content') ?>
    </div>
    <!-- end center -->
    <!-- start right -->
    <div class="col-right side-col">
      <?php echo $this->getChildHtml('right') ?>
    </div>
    <!-- end right -->
  </div>
</div>
<!-- end middle -->
<!-- start footer -->
<div class="footer-container">
  <div class="footer">
    <?php echo $this->getChildHtml('footer') ?>
  </div>
</div>
<!-- end footer -->
```

Andere Webapplikationen binden nacheinander die Templates ein, um so die (X)HTML-Ausgabe zusammenzubauen. *Magento* bündelt stattdessen sämtliche Seiteninhalte und vereinfacht die Anordnung mittels Blöcken.

Elemente des Magento-Themas

Positiver Eindruck beim Shop-Besucher

Ein Online-Shop muss in den Augen eines Betrachters, also Ihres Shop-Besuchers, einen positiven Eindruck hinterlassen. Soll ein Besucher zu einem zahlenden Kunden werden, dann muss emotional, inhaltlich und technisch alles stimmen: Artikelangebot, **Produktdarstellung**, Artikel-

informationen, Produktbilder, Preisniveau, Zahlungsarten, Usability und vieles mehr. Aus der Sicht eines Shop-Betreibers versuchen Sie dem potenziellen Kunden so viel wie möglich davon zu bieten, um seine Wünsche und Bedürfnisse bestmöglich zu befriedigen. Sie haben es in der Hand, denn Sie treffen die Artikelauswahl, pflegen Produktinformationen und stellen die **Technik** (Soft- und Hardware). Nur ein zufriedener User kann zum Käufer werden und künftig vielleicht zum Stammkunden.

Die besonderen Vorteile bei *Magento* im Designbereich sind:

- Maximale Anpassbarkeit: Designänderungen sind nicht nur auf jeder einzelnen CMS-Seite möglich, sondern sogar auf jeder Kategorie- und Produktebene. Dadurch verbessern Sie im Shop die Verkaufschancen anhand verkaufsförderlicher Marketing-Maßnahmen, z. B. **Up-/Cross-Selling**.
- Minimale **debugging time**: Das Shop-System reduziert die Zeiten bei der Suche nach Fehlern. Das modulare Template-System verringert die Menge an (X)HTML-Code. Weniger Quellcode führt in **Validatoren** zu weniger Fehlern.
- Multiple Themen: Die Software bringt die Fähigkeit mit, mehrere Themen parallel zu nutzen. Im Backend gibt es die Möglichkeit, zwischen diversen Layouts teils sogar **termingesteuert** hin- und herzuwechseln.
- Kontinuierlicher Arbeitsablauf: Durch die objektorientierte Programmierung von *Magento* greifen Sie in jeder Template-Datei direkt auf **Template Tags** zu. Viele Aufgaben wie das Ein- und Ausblenden von Modulen sind einfach im Backend möglich, ohne dass Sie jedes Mal einen Programmierer beauftragen.

Product Tags, Template Tags und Block Tags

Neue Produktschlagworte hinterlegen

Product Tags sind **Produktschlagwörter**. Registrierte Shop-Besucher bekommen die Möglichkeit, im Frontend auf der Produktdetailseite eigene Produktschlagwörter zu hinterlegen. Diese Tags helfen Kunden dabei, sich besser an Artikel zu erinnern. Auf der *Magento*-Homepage finden Sie diese nützliche Funktion bei den Paketerweiterungen. Zudem gibt es eine Schnellnavigation im Shop zu getagten Artikeln mit bereits

vorhandenen Product Tags, die sogenannte **Tag Cloud** für beliebte Schlagwörter. Die Zahl in Klammern signalisiert, wie häufig dieses Schlagwort bisher genutzt wurde. Im Backend-Bereich finden Sie alle Tags gebündelt unter »Katalog › Schlagworte« (Catalog › Tags). Hier aktivieren Sie als Shop-Betreiber die Tags und schalten diese damit für den Frontend-Bereich frei.

Abbildung 3.5: Neue Produktschlagwörter hinzufügen

Block Tags und Template Tags

In Verbindung mit der Designentwicklung sind die **Template Tags** viel spannender. Zwar ist auf den CMS-Seiten und den statischen Block-Seiten des Shop-Systems kein PHP-Code möglich, aber mit Hilfe einiger vordefinierter Tags gelingt dies dann doch. Diese Tags erkennen Sie an den Klammern `{{ … }}`, die dann ein **Parser** durch den entsprechenden Quellcode ersetzt. Einige Tags finden Sie beispielsweise in den Template-Dateien. Bei diesen Templates beinhalten die HTML-Dateien die sogenannten Template Tags, die für die Anzeige verschiedenster Inhalte verantwortlich sind. Ähnlich funktionieren **Block Tags**, mit denen Sie auf dynamische Inhalte zugreifen, die Sie auf statischen Blöcken oder CMS-Seiten anzeigen.

Einige kurze Code-Beispiele demonstrieren Ihnen das viel anschaulicher:

Listing 3.2: Beispiele für Block Tags und Template Tags

```
# Bindet innerhalb einer CMS-Seite den Inhalt einer anderen an
{{block type="cms/page" page_id="kundenservice"}}
# Stellt die Tag Cloud auf einer beliebigen CMS-Seite dar
{{block type="tag/popular" template="tag/popular.phtml"}}
# Zeigt Newsletter-Anmeldung auf CMS-Seite oder statischen Block
{{block type="core/template" name="contactForm" template="newsletter/
subscribe.phtml"}}
# Präsentiert auf einer Seite ein Bild
<img src="{{skin url='images/bild.png'}}"
     alt="" width="" height="" />
# Erstellt einen Link zur Startseite
<a href="{{store url=""}}">Home</a>
```

Tipp

Mehrere Formulare auf einer Seite

*Binden Sie nicht zu viele Formulare auf einer einzelnen Seite ein. Das normale **Suchformular** (**GET-Methode**), die **erweiterte Suche** (GET-Methode), die **Newsletter-Anmeldung** (POST-Methode) und das **Kontaktformular** (**POST-Methode**) basieren alle auf Formularen. Möglicherweise funktionieren sonst die Formulare nicht korrekt. Setzen Sie auf einer Seite mehrere Formulare, dann verwenden Sie im Bedarfsfall unterschiedliche Formularnamen und JavaScript-Funktionen. Am besten beschränken Sie sich pro Seite auf maximal zwei Formulare.*

Als Nächstes zeigen wir Ihnen exemplarisch, wie Sie ein neues Werbebanner auf der Homepage platzieren. Sie erledigen das im Handumdrehen:

Step

1. Erstellen Sie einen statischen Block mit neuem HTML-Inhalt.
2. Fügen Sie im gewünschten Haupt-Template die Block-Ausgabe ein.
3. Editieren Sie auf der CMS-Seite Ihrer Homepage das kundenspezifische Layout.

Werbebanner als Block einbauen

Zunächst fügen Sie bei »CMS › Statische Blöcke« (CMS › Static Blocks) einen neuen statischen Block hinzu. Diesem geben Sie den Titel **Werbebanner** und den **Seitenbezeichner** (**Identifier**) werbebanner. Merken Sie sich diesen Seitenbezeichner, da Sie ihn später noch einmal benötigen. Vergessen Sie nicht, vor dem Speichern den Status auf aktiv (enabled) zu setzen. Im Inhalts- bzw. Content-Bereich hinterlegen Sie den Verweis auf Ihr Werbebanner-Bild: `<img src="{{skin url='images/werbebanner.png'}}" alt="" width="" height="" />`. Geben Sie auf jeden Fall noch die beiden Attribute `width` und `height` mit, welche die gewünschte Bildgröße liefern. Denn es gibt zwei Arten, wie Browser ein Bild darstellen. Einige beziehen sich auf die tatsächliche physikalische Größe, während sich andere an den Werten orientieren, die Sie selbst im Quelltext hinterlegen. Der größte Teil ist ganz normaler HTML-Quellcode. Mit dem Template Tag verweisen Sie jedoch, wie Sie sicherlich fast vermuten, auf den Pfad `/skin/frontend/default/default/`.

Startseite bearbeiten

Über die CMS-Seitenverwaltung erkennen Sie, dass Ihre Startseite als Layout die Datei 2columns-right.phtml einsetzt. Diese und einige andere Layoutdateien finden sich unter `/app/design/frontend/default/default/template/page`. Öffnen Sie die Datei und fügen Sie den Quellcode-Schnip-

sel <?php echo $this->getChildHtml('banner') ?> an der Stelle ein, wo das Banner später erscheinen soll. Zu guter Letzt editieren Sie bei den CMS-Seiten Ihre Startseite. Gehen Sie in das Feld »Eigene Gestaltung« (Custom Design) und fügen Sie dort im Feld »XML für Layoutänderung« (Layout Update XML) den in Listing 3.3 dargestellten Quellcode ein. So verknüpfen Sie den as-**alias** banner mit dem Identifier werbebanner. Die Startseite weiß nun, welchen Inhalt sie anzeigen soll.

Listing 3.3: Alias mit Identifier verknüpfen

```
<reference name="root">
  <block type="cms/block" name="banner" as="banner">
    <action method="setBlockId"><id>werbebanner</id></action>
  </block>
</reference>
```

Action-Methoden nutzen

Für die Arbeit mit Template Tags gibt es noch weitere **Action-Methoden**, mit denen Sie Layouteinstellungen ändern, ohne an den *.phtml-Dateien etwas zu berichtigen. Die wichtigsten Methoden haben wir für Sie in Tabelle 3.1 gesammelt. Wenn Sie die XML-Dateien im Pfad /app/design/frontend/default/default/layout näher begutachten, erhalten Sie bestimmt noch einige Anregungen. Die Auswirkungen sehen Sie, sobald Sie Ihren Shop über den Browser öffnen oder den entstandenen Quellcode-Source betrachten.

Methode	Aufgabe der Parameter
addCss addJs	Fügt auf der Seite eine zusätzliche StyleSheet-/JavaScript-Datei ein <link>style.css</link> bzw. varien/script.js
addCssIe addJsIe	Gibt eine CSS-/JS-Datei aus, wenn ein spezieller Browser genutzt wird <link>style.css</link> bzw. varien/script.js
addLink	Setzt eine Einstellung, auf die Sie in den Template-Dateien zugreifen <name>account</name> <path>customer/account/</path> <label>Account Dashboard</label>
setBlockId	Zeigt auf der Seite einen statischen Block an <id>werbebanner</id>
setCartTemplate setEmptyTemplate chooseTemplate	Setzen der Template-Datei abhängig von der Artikelzahl im Warenkorb <value>checkout/cart.phtml</value> <value>checkout/cart/noItems.phtml</value>

Tabelle 3.1: Einige beliebte Action-Methoden

Methode	Aufgabe der Parameter
setCharset	Überschreibt die standardmäßige Kodierung einer Seite (Standard: UTF-8)
	`<charset>ISO-8859-2</charset>`
setContentType	Überschreibt den Standard-Header text/html einer Seite
	`<content>text/xml</content>`
setTemplate	Wechselt die Template Datei auf einer einzelnen Seite
	`<template>page/3columns.phtml</template>`

Tabelle 3.1: Einige beliebte Action-Methoden (Forts.)

Begriffe und Grundlagen zum Layout

Flash-Videos einbinden

Wie bereits erwähnt, spielen beim Erstellen eines maßgeschneiderten Layouts die beiden Verzeichnisse `/app` und `/skin` eine wichtige Rolle. Wenn Sie für Ihren Shop ebenfalls ein individuelles Design basteln möchten, dann müssen Sie sich mit den beiden Ordnern vertraut machen. Im ersten Themen-Ordner innerhalb eines Interface liegen Layout-, lokale Übersetzungs- und Template-Dateien. Das zweite Verzeichnis beinhaltet Bilder sowie blockspezifische JavaScript- und Cascading-Stylesheet-Dateien. Natürlich lassen sich genauso andere Dokumenttypen hier ablegen, beispielsweise PDF-Dateien. Etwas komplizierter, aber durchaus möglich ist das Einbinden von Flash-Videos. Solche **SWF-Dateien** legen Sie ebenso im `/skin`-Verzeichnis ab, als Vorschlag nutzen Sie den Ordner `/images/media/`. Eine externe Lösung zum Abspielen von Videos ist *YouTube*, als interne Lösung benötigen Sie einen **Flashplayer** wie `flowplayer.org`.

WWW.....

`magentocommerce.com/wiki/flash`
MagentoCommerce Wiki (Flash-Datei auf Startseite einbinden)

`magentocommerce.com/wiki/groups/248/display_flash_videos_in_the_product_gallery`
MagentoCommerce Wiki (Flash-Videos in Produktgalerie anzeigen)

`mxperts.de/youtube-videos-in-magento-einbinden/`
mxperts (YouTube-Videos in Magento einbinden)

Sicherheit der Shop-Dateien

Diese beiden Hauptverzeichnisse im root-Verzeichnis Ihres Shops bilden daher den wichtigsten Ausgangspunkt beim Erstellen eines neuen Themas. Natürlich hat die Trennung in diese beiden Verzeichnisse auch einen Grund. Aufgrund gewisser Sicherheitsaspekte lassen sich die Dateien im `/app`-Ordner komplett vor Shop-Besuchern verstecken. Die

Dateien im `/skin`-Verzeichnis sind dagegen weniger sicherheitsrelevant und frei aus dem Web zugänglich. Dies unterstützt zusätzlich die **Sicherheitsbelange** Ihres Shops. Vernachlässigen Sie jedoch nicht die persönlichen Sicherheitsbelange, denn Ihre eigenen Bilder, Stylesheets oder Skripte benötigen ebenso Schutz vor fremden Zugriffen. Bei Bedarf sichern Sie diese zusätzlich mittels htaccess-Datei ab.

Bevor Sie erste Anpassungen vornehmen, sollten Ihnen in beiden Ordnern mehrere scheinbar doppelte Verzeichnisse mit dem Namen default auffallen:

- Verzeichnis: `/app/design/frontend/default/default/`
- Verzeichnis: `/skin/frontend/default/default/`

Interface- und Themen-Name

In beiden Fällen bezeichnet der erste default-Ordner den Interface-Namen und der zweite default-Ordner den Namen des Themas. Sie können selbst weitere Interfaces und Themen anlegen. Entscheidend hierbei ist, dass Sie in beiden Hauptordnern identische Namen vergeben. Nicht gleich bezeichnen müssen Sie in einem Verzeichnis die beiden Ordnernamen für **Interface** und **Thema**, z. B. `/myinterface/mytheme/`. Innerhalb eines Interface-Verzeichnisses dürfen Sie beliebig viele Themen-Ordner anlegen. Allerdings kann ein *Magento*-Store momentan nur das default-Thema und ein weiteres höher priorisiertes Thema laden. Die Fähigkeit, beliebig viele Themen zu laden, soll in einer künftigen Version verfügbar sein.

Hierarchie von Themen

Logische Struktur der Dateien

Bevor wir damit fortfahren, ein neues Thema anzulegen, möchten wir Sie über die hierarchische Strukturierung von Themen informieren. Mit diesem Wissen ausgestattet verstehen Sie besser, welche Dateien Sie später benötigen, um ein neues eigenes Thema anzulegen. Die Entwickler des Shop-Systems haben besonderen Wert darauf gelegt, dass die Webanwendung fehlerfrei läuft und jederzeit alle für das Aussehen zuständigen Dateien finden und laden kann. Für die einwandfreie Funktionsweise mehrerer parallel vorhandener Themen spielt einerseits die logische Strukturierung der Dateien eine entscheidende Rolle. Andererseits muss die Technik für den Shop-Betreiber bzw. dessen Designer so einfach und verständlich wie möglich sein.

Das Thema mit der niedrigsten Hierarchieebene ist momentan das default-Standardthema. Eine höhere Hierarchiestufe hat nur das von Ihnen im Admin-Bereich einem Store zugewiesene Thema auf Website- oder Store View-Ebene. Ruft ein Online-Shop-Besucher eine Seite im Shop auf, dann ruft er beispielsweise die Template-Datei des Kategorie-Listings `/template/catalog/category/view.phtml` auf. Ist die Shop-Anwendung in Ihrem Design nicht in der Lage, diese Template-Datei zu finden, versucht es, diese Datei von einer niedrigeren Hierarchiestufe zu laden. Gelingt dies nicht, versucht das System auf der nächstniedrigeren Stufe die Datei view.phtml aufzuspüren. Dies setzt sich so lange fort, bis das System fündig wird und die Dateisuche beenden kann. Aufgrund der beschriebenen Vorgehensweise bekommt die angewandte Methode zur Gewinnung eines Designs den Namen **Fallback**.

default	mytheme1	mytheme2
Alle anderen Dateien		
/css/base.css	/css/base.css	
/css/boxes.css	/css/boxes.css	
/images/logo.gif	/images/logo.gif	
/templates/ 3-col-layout.phtml	/templates/ 3-col-layout.phtml	
/templates/ header.phtml	/templates/ header.phtml	
/css/base.css		/css/base.css
/templates/ 3-col-layout.phtml		/templates/ 3-col-layout.phtml

Tabelle 3.2: Drei Themen für das Shop-Design

Identische Dateien sind redundant

Tabelle 3.2 enthält für einen Store drei Themen. Das default-Standard-Thema beinhaltet logischerweise alle erforderlichen Dateien. Individuelle Themen, wie mytheme1 und mytheme2, haben prinzipiell immer die höchste Hierarchiestufe. Die niedrigste Stufe bekommt das Standard-Thema default. Bei genauem Hinsehen werden Sie feststellen, dass die beiden Dateien `3-col-layout.phtml` und `base.css` mehrfach vorhanden sind. Falls die Dateien inhaltlich identisch sind, löschen Sie besser die Dateien, da diese redundant sind. Handelt es sich bei den Dateien um unterschiedliche Dateien oder Versionen, so sind die Dateiinhalte nicht redundant.

Die oben beschriebene Fallback-Methodik durchläuft in diesem Fall die Tabelle von rechts nach links. Je nachdem, in welchem Store sich der User befindet, verwendet er ein bestimmtes Thema. Findet das System in diesem höher priorisierten Thema die nötige Datei, wird diese Information ausgegeben, andernfalls geht das System über auf die Datei des default-Verzeichnisses. Hier wird die Suche auf jeden Fall immer fündig und daher erfolgreich beendet. *Magento* lädt zunächst das Standard-Thema und stülpt anschließend das individuelle Thema über das zuvor geladene. Wenn Sie alle Dateien in Ihr eigenes Thema kopieren, benötigt das Laden des Layouts nur unnötig viel Zeit.

Themen für Haupt- und Nebensaison

Mit dem Shop-System betreuen Sie mehrere Themen gleichzeitig (**Multiple Themes**). Sobald Sie ein eigenes Thema erstellen, arbeiten Sie künftig parallel immer mit mindestens zwei Themen, dem Standard- und dem Nicht-Standard-Thema. Für Sie als Shop-Betreiber ist es ein leichtes Unterfangen, von einem Thema auf ein anderes umzuschalten. Sie haben in der **Nebensaison** immer ein Standard-Thema griffbereit, während Sie im Hintergrund weitere Themen anlegen und bearbeiten. Die zusätzlichen Themen kommen zu speziellen Gelegenheiten oder in der **Hauptsaison** zum Einsatz, wie Weihnachten, Frühlingsbeginn, Halloween, Urlaubszeit, Ostern, Valentinstag …

Ein maßgeschneidertes Thema sieht zur Weihnachtszeit bevorzugt etwas anders aus. Das Potenzial der saisonabhängigen Einkäufe nutzen Sie besser, wenn Sie das Aussehen Ihres Shops an die Wünsche und bevorstehenden Anlässe anpassen. Zum Glück geht das mit *Magento* wesentlich schneller als bei vergleichbaren Layoutlösungen. Alles, was Sie vor **Weihnachten** benötigen, ist etwas mehr rote Farbe, Schneeflocken und ein Weihnachtsmann.

Nichtstandardisiertes Thema

Der designtechnische Unterschied in Abbildung 3.6 entsteht in den beiden Shops mit wenigen Dateien. Sie ändern nur ein paar CSS-Dateien, Hintergrundbilder, Grafiken, Icons und eventuell einige Textstellen. Weil Sie dafür an relativ wenigen Stellen Änderungen vornehmen müssen, brauchen Sie nicht ein komplett neues Template anzulegen. Es ist völlig ausreichend, ein paar Dateien in einem nichtstandardisierten Thema abzulegen.

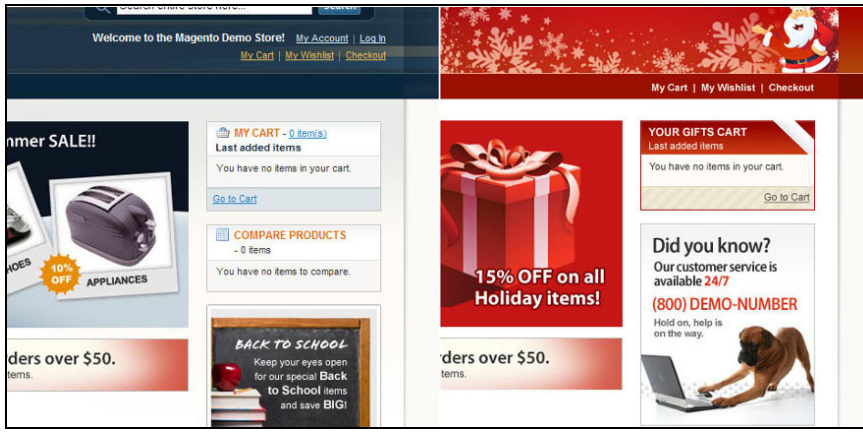

Abbildung 3.6: Multiple Themen zu Weihnachten

Tipp

Eigenes Logo in Magento einbinden

Der Standardbilderordner heißt images. Für die eigenen Logos legen Sie im Verzeichnis `/skin/frontend/default/default/` *den Ordner »logos« an und kopieren Ihr Logo dort hinein. Im Backend-Bereich öffnen Sie dann die Sektion »Kopfzeile« (Header) unter »System › Konfiguration › Gestaltung« (System › Configuration › Design). Dann ändern Sie nur noch den Bildpfad zu »logos/logo.png«.*

Individuelles Thema erstellen

Erste Schritte zum eigenen Thema

Nachdem Sie inzwischen ein besseres Verständnis davon haben, wie sich ein Thema zusammensetzt, ist es nicht mehr schwer, selbst ein individuelles Thema zu erstellen. Da die grundlegende Struktur bereits beschrieben wurde, zeigen wir Ihnen auf den folgenden Seiten die einzelnen Schritte:

Step

1. Duplizieren Sie das bestehende Standard-Thema.
2. Kopieren und bearbeiten Sie einzelne Dateien für das individuelle Thema.
3. Ändern Sie im Backend-Bereich den Themen- und Interface-Namen.

Ordnernamen festlegen

Im ersten Schritt benötigen Sie ein **Duplikat** des Standard-Themas mit allen darin enthaltenen Verzeichnissen und Dateien (Abbildung 3.3). Duplizieren Sie dazu die Inhalte der beiden Ordner `/app/design/frontend/` und `/skin/frontend/`. Sämtliche darin enthaltenen Dateien steuern, wie

der Ordnername schon andeutet, das Aussehen des Frontends. Innerhalb der frontend-Ordner finden Sie neben der standardmäßig vorhandenen Ordnerstruktur `/default/default/` dann beispielsweise das neue Interface und Thema `/myinterface/mytheme/`. Nutzen Sie für neue Ordnernamen bevorzugt **alphanumerische Kleinbuchstaben** von a bis z und **Unterstriche**, wobei der Name nicht mit einer Zahl beginnen soll, z. B. `/myinterface2009/theme_blue/`.

Anhand der Verzeichnisnamen erkennt das Shop-System die Bezeichnungen für Ihr Interface und Ihr Thema. Diese Information brauchen Sie für die Zuordnung des eigenen Themas im Backend-Bereich. Sobald Sie dies erledigt haben, besitzt Ihr Shop ein neues eigenes Standard-Thema, das Sie als Basis für die Erstellung Ihres eigenen Themas nutzen. Wenn Sie den Abschnitt über die **Hierarchie** von Themen gelesen und verstanden haben, dann wissen Sie, was Sie als Nächstes machen müssen, um ein neues Non-Default-Thema zu erstellen. Das ist ganz einfach: nichts.

Spezifische Anpassungen

Sie müssen keine vorhandenen Verzeichnisse oder kompletten Teile des Standard-Themas duplizieren. Sie nehmen künftig nur spezifische Anpassungen an wenigen ausgewählten Dateien vor. Wenn Sie Dateien verändern wollen, holen Sie sich die identifizierte Datei aus dem Standard-Thema und verwenden diese als Vorlage. Nur an dieser Kopie nehmen Sie Ihre Änderungen vor. Sie müssen nur eine einzige Regel beachten bei der Arbeit mit individuellen Themen: Behalten Sie die Verzeichnisstruktur bei, die das Standard-Thema besitzt.

Ein praktisches Beispiel soll Ihnen verdeutlichen, was wir damit meinen. Nehmen wir an, Sie wollen Änderungen an der Template-Datei form.phtml vornehmen, die sich innerhalb des Kontakt-Moduls befindet. Mit dieser Datei ändern Sie das Aussehen des Kontaktformulars. Die Originaldatei aus dem app-Zweig, die Sie als Vorlage öffnen, liegt im Verzeichnis `/template/contacts/`. Erstellen Sie in Ihrem individuellen Thema die gleichen Ordner `/template/contacts/` und legen Sie eine Kopie der Datei form.phtml dort hinein. An dieser Datei nehmen Sie nun die gewünschten Änderungen vor. Erst wenn die Ordnerstruktur identisch aussieht, sehen Sie die vorgenommenen Anpassungen im Frontend.

Zeitgesteuerte Themen nutzen

Sind Sie mit der neuen Optik fertig, dann aktivieren Sie unter »System › Konfiguration › Gestaltung« (System › Configuration › Design) das geänderte Thema. Für die CMS-Seiten bei »CMS › Seiten verwalten«

(CMS › Manage Pages) gibt es sogar die Möglichkeit, **zeitgesteuert** Themen anzuzeigen. In Abbildung 3.7 sehen Sie eine beispielhafte Konfiguration für die Weihnachtszeit.

Abbildung 3.7: Zeitgesteuertes Thema auf der Startseite im CMS

Das erste eigene Design

Template-Aufbau neu definiert

Wer schon mit anderen eCommerce-Lösungen gearbeitet hat, der muss zuerst einmal etwas umdenken bzw. umlernen. Keine Shop-Software arbeitet wie die andere. Das gilt genauso für *Magento* – es gibt zunächst viele neue Features und Techniken zu entdecken. Doch wenn Sie sich daran gewöhnt haben, dann kommen Sie künftig schneller und oft bequemer ans Ziel. Einige Arbeitsabläufe sind sicherlich anders als vorher. Scheuen Sie aber nicht die schwierige Lernphase, nebenbei stoßen Sie auf eine ganze Reihe neuer Tricks, die Ihnen den Betrieb Ihres Shops wesentlich erleichtern.

Bleiben Sie am Ball, beißen Sie sich durch. Sie werden belohnt mit wachsenden Verkaufszahlen. Beginnen Sie jetzt und hier mit der Arbeit an Ihrem ersten eigenen Layout. Schritt für Schritt entsteht das Design. Doch selbst zu dem Zeitpunkt, wo Sie den Shop online einstellen, sind Sie noch lange nicht fertig. Es gibt immer etwas zu tun oder zu verbessern. Nicht nur das Layout, sondern auch die enthaltenen Artikel und die eingebunden Funktionen werden sich verändern, genauso wie die Umsätze steigen.

Unterschied: Blöcke und Layouts

Bevor Sie mit dem Design loslegen, nehmen Sie sich noch kurz Zeit für ein paar weitere hilfreiche Zusammenhänge. Bei der Designentwicklung für *Magento* ist es gut, die Kombination aus Blöcken und Layout zu verstehen. In der Abbildung 3.8 sehen Sie eine kleine Merkhilfe, wie sich aus einzelnen Blöcken ein langer Gesamtblock bildet:

- Bild 1: Stellen Sie sich ein leeres Grundgerüst ohne Inhalt vor.
- Bild 2: Füllen Sie das leere Grundgerüst mit dem ersten Block.
- Bild 3: Fügen Sie unter dem ersten Block einen zweiten Block ein.
- Bild 4: Dehnen Sie das Grundgerüst mit einem dritten Block weiter aus.
- Bild 5: Sehen Sie, wie das Grundgerüst mit dem vierten Block weiter wächst.

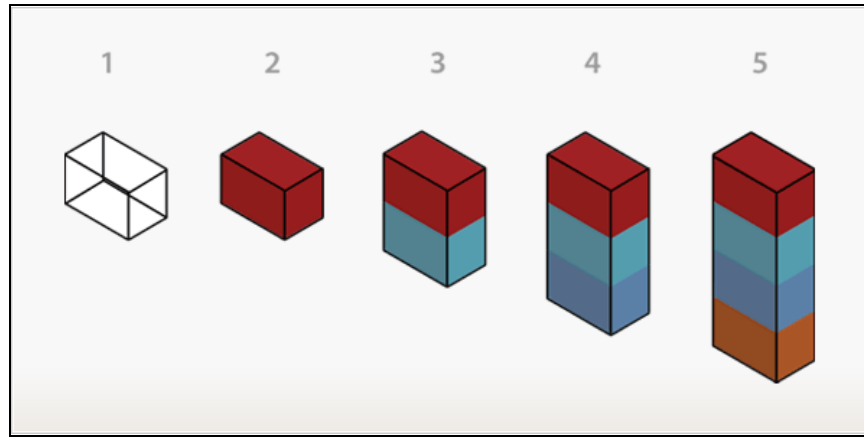

Abbildung 3.8: Spaltenbildung aus einzelnen Blöcken

Seitenstruktur und Block-Position

Betrachten Sie nun die **Landing Page** von Shirts in Ihrem Shop mit den Testdaten. Danach schauen Sie sich die Aufteilung der beiden rechten Einzelbilder in Abbildung 3.9 an. Links sehen Sie den Original-Screenshot, im mittleren Bild die grobe Seitenstruktur und rechts außen die einzelnen Blöcke. Die obige Merkhilfe, verglichen mit den realen Bildern, symbolisiert das gesamte Layout von der Planung bis hin zur Umsetzung. Die grobe **Seitenstruktur** definiert die Grundstruktur Ihrer Webseite: Header-, Main-Content-, Footer-, linker und rechter Spaltenbereich. Diese Bereiche dienen als Grundgerüst für die **Positionierung** der Content-Blöcke innerhalb der Seite. Mit den einzelnen Content-Blöcken realisieren Sie die verschiedenen Funktionalitäten auf einer Seite, wie Kategorieliste, Callout, Mini-Warenkorb, Product Tags etc. Erstellen Sie dann den (X)HTML-Code mittels der Template-Dateien.

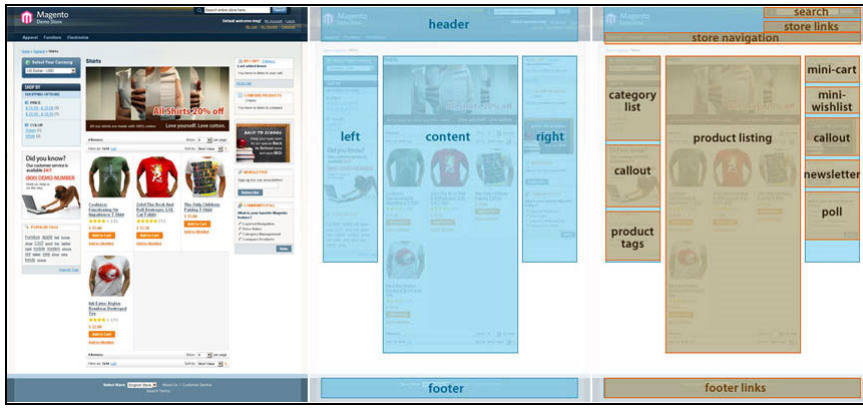

Abbildung 3.9: Seitenstruktur und Content-Blöcke einer Landing Page

Mit dem **Layout** positionieren Sie die Content-Blöcke im Grundgerüst der Seitenstruktur. Das Layout existiert in Form von **XML-Dateien**, was auf den ersten Blick etwas ungewöhnlich erscheint. Im Verzeichnis `/app/design/frontend/default/default/layout` finden Sie diese Dateien. Mit diesen Layoutdateien verschieben Sie die Blöcke auf einer Seite und weisen den kleineren Content-Blöcken mittels **Action**-Methoden (z. B. setTemplate) ein Template zu. Bereits mit wenigen Handgriffen an Layout- und Template-Dateien verändern Sie die visuelle Optik Ihres Webauftritts. Nähere Informationen über Layouts finden Sie am Ende dieses Kapitels.

Arbeitsablauf der Designentwicklung

Blöcke, Templates, Layouts und Skins

Nachdem Sie inzwischen mit allen relevanten Wissensgebieten rund um das Design-Thema vertraut sind, können Sie sich selbst daran machen, ein Layout zu entwickeln. Den Entwurf eines Designs unterstützen die folgenden Tools und Dateien: Blöcke, Layouts (*.xml), Templates (*.phtml), Skins (*.css, *.gif, *.jpg, *.png und *.js). Jetzt starten wir mit den ersten Schritten zur Erstellung eines Designs. Folgende Arbeitspakete bestimmen den Arbeitsablauf:

Step.....

1. Duplizieren Sie das bestehende Standard-Thema.
2. Aktualisieren Sie die Frontend-Anzeige durch einen Cache-Refresh.
3. Bestimmen Sie die strukturellen Blöcke auf den einzelnen Seiten.
4. Erstellen Sie für jeden einzelnen Block eine Template-Datei.
5. Verändern Sie in der Layoutdatei die Modulausgabe und die Modulposition.

Bevor Sie mit der Bearbeitung einzelner Dateien beginnen, benötigen Sie ein eigenes Thema, an dem Sie Änderungen vornehmen, ohne das Grundlayout zu überschreiben. Im Notfall haben Sie so die Möglichkeit, jederzeit im Backend auf das ursprüngliche unveränderte Standardlayout umzuschalten. Sind Ihre Layoutanpassungen in den neuen Dateien fehlerhaft, dann greifen Sie auf die Originaldateien zu und kopieren den funktionsfähigen Quellcode heraus. Daher empfehlen wir Ihnen, gleich zu Beginn das Standardlayout zu duplizieren. Wie Sie dazu vorgehen, zeigen wir Ihnen am Ende von Kapitel 3.

Cache deaktivieren

Nehmen Sie viele kleine Designanpassungen vor, dann lohnt es sich, den Cache komplett abzuschalten. Wählen Sie dazu in der **Cache-Verwaltung** (Cache-Management) bei »Alle Caches« (All Caches) im Drop-down-Feld den Punkt »Deaktivieren« (Disable). Ihr Shop übernimmt die Konfiguration, sobald Sie auf »Cache-Einstellungen speichern« (Save cache settings) klicken. Durch diese geänderte Einstellung schaltet das System alle Cache-Aktivitäten ab. Der Vorteil ist: Sobald Sie an irgendwelchen Layout- oder Template-Dateien Änderungen abspeichern, sehen Sie nach einem **Refresh** des Browsers mit F5 sofort, ob die Änderungen korrekt funktionieren. Sie sparen sich dadurch jedes Mal das manuelle Aktualisieren (Refresh) des Cache. Denken Sie daran, nach getaner Arbeit den Cache zur Beschleunigung der Shop-Zugriffe wieder einzuschalten.

Ist-Zustand festhalten

Bevor Sie mit dem neuen Design am Quellcode loslegen, erfassen Sie zunächst den Ist-Zustand. Machen Sie sich die Mühe und identifizieren Sie alle Seiten Ihres Shops. Das mag auf den ersten Blick ziemlich aufwändig erscheinen, ist es aber nicht wirklich. Selbst ein Shop mit vielen Tausend Artikeln besteht zumeist aus weniger als 100 Seiten, denn eine Produktdetailseite sieht im Normalfall (innerhalb einer Kategorie) immer gleich aus. Ein kleiner Shop kommt deshalb schon mit 10 bis 20 Seiten aus. Legen Sie sich einfach eine kleine *Excel*-Tabelle an. Dies ist eigentlich kein Vorschlag, sondern ein Muss, da es die Arbeit erheblich erleichtert. In Tabelle 3.3 sehen Sie einen Auszug, ohne die Blöcke, die auf den einzelnen Seiten zum Einsatz kommen. Notieren Sie darin den Seitennamen, die Template-Datei, die verwendeten Blöcke und vorbereitend für später die **Keywords** für die Suchmaschinenoptimierung. Bei Artikeln und Kategorien arbeiten Sie bei den Keywords natürlich mit **Variablen**, wie `$artikelname` oder `$kategoriebezeichnung`. Der Vorteil einer solchen Liste ist, dass Sie sofort wissen, auf welchen Seiten die Änderungen auftreten.

Seitenname	Skeleton Template	Keywords
Homepage	2columns-right.phtml	Electronics, Apparel, Furniture
Landing Page	1column.phtml	Electronics Shop
Kategorie-Liste	3columns.phtml	Digital Cameras, Cameras
Produkt-Detailseite	2columns-right.phtml	Kodak EasyShare Digital Camera
Impressum	3columns.phtml	Impressum Shop
Kundenservice	3columns.phtml	Kundenservice Shop

Tabelle 3.3: Seitenstruktur mit Templates und Keywords (ohne Blöcke)

Vorhandene Dateien verwenden

Alle bereits vorhandenen Template-Dateien, die standardmäßig für die Seitenstrukturierung genutzt werden können, befinden sich im Verzeichnis `/app/design/frontend/myInterface/myTheme/template/page/`. Die Dateien, die Sie dort finden, nennt man auch **Skeleton** Templates, da sie das Gerippe bzw. **Grundgerüst** einer Seite bilden. Ihre Aufgabe besteht lediglich darin, die strukturellen Blöcke auf der Seite mittels `<div>`-Container zu positionieren.

Listing 3.4: Stark vereinfachtes Beispiel eines Skeleton Template

```
<body>
  <div class="wrapper">
    <div class="header">
      <?php echo $this->getChildHtml('header') ?>
    </div>
    <div class="middle">
      <?php echo $this->getChildHtml('breadcrumbs') ?>
      <div class="col-left side-col">
        <?php echo $this->getChildHtml('left') ?>
      </div>
      <div id="main" class="col-main">
        <?php echo $this->getChildHtml('content') ?>
      </div>
    </div>
    <div class="footer">
      <?php echo $this->getChildHtml('footer') ?>
    </div>
  </div>
</body>
```

Block-Identifier im Einsatz

Das einzig Auffällige an diesen speziellen Template-Dateien ist der mehrfach vorkommende PHP-Methodenaufruf `$this->getChildHtml()`. Mit einem solchen Aufruf lädt und positioniert *Magento* die strukturellen Blöcke

und stellt deren Inhalte auf der Seite dar, auf der dieses Grundgerüst eingesetzt wird. Der Name in Klammern bezeichnet jeden einzelnen Strukturblock innerhalb des Layouts, die Block-**Identifier** in Listing 3.4 lauten header, breadcrumbs, left, content und footer. Die Zuweisung der Skeleton Templates zum Store erfolgt über das Layout.

Tipp . . . . . .

Layoutentwurf mit Wireframes

Für einen ersten konzeptionellen Layoutentwurf Ihres Shops können ***Wireframes*** *eine gute Hilfe sein. Viele Designer entwerfen die Optik direkt mit Freehand, PhotoShop oder Dreamweaver (ehemals von Macromedia). Doch für einen ersten Entwurf reicht oftmals eine Skizze aus, so ähnlich wie in Abbildung 3.4. Anfangs geht es meist um banale Dinge, wie Reihenfolge der Boxen bzw. zweispaltig oder dreispaltig. Dafür brauchen Sie oft keinen teuren Webdesigner. Erst für die farbliche Gestaltung lohnt sich der Gang zum Profi. Versuchen Sie es einfach mal selbst, in Kapitel 3.3 zeigen wir Ihnen, mit welchem Tool Sie so etwas umsetzen. Das Ergebnis ist dann gleichzeitig die Grundlage für die ersten Layoutanpassungen Ihres Online-Shops. Zudem sind solche Wireframes eine wertvolle Unterstützung für jeden externen Programmierer.*

Blöcke benötigen Template-Dateien

Nachdem Sie die einzelnen sich wiederholenden Blöcke mittels *Excel*-Tabelle (in Textform) oder Wireframes (in Bildform) identifiziert haben, kommen die Template- Dateien an die Reihe. Jeder Block, den Sie optisch verändern oder neu erstellen, benötigt ein eigenes (Mini-)Template, welches das Aussehen, den Inhalt und die Formatierung festlegt. Bei neuen Dateien empfehlen wir Ihnen möglichst beschreibende und aussagekräftige Dateinamen. Wie Sie in Abbildung 3.9 sehen, kann eine einzige Seite locker aus 15 Blöcken bestehen. Für jeden Block, den Sie in Ihrem neuen Layout einbauen, benötigen Sie eine eigene Template-Datei. Zum Glück enthält das Standard-Thema bereits eine umfassende Sammlung von mehr als 30 Vorlagen für die wichtigsten Anforderungen. Sie brauchen das Rad also nicht neu zu erfinden, sondern nutzen für Ihren Designentwurf die vorhandenen Template- und CSS-Dateien.

Design-Hilfe aktivieren

Eine wesentliche Arbeitserleichterung für die Designentwicklung stellen die »**Vorlagen Pfadhinweise**« (Template Path Hints) und »Blocknamen zu Hinweisen« (Add Block Names to Hints) dar. Damit Sie diese Design-Hilfe aktivieren können, wählen Sie im Backend-Bereich »System › Konfiguration › Entwickleroptionen« (System › Configuration › Developer).

Wählen Sie zuerst im aktuellen **Konfigurationsbereich** (Current Configuration Scope) die gewünschte Store View aus, denn ansonsten ist es nicht möglich, die hilfreichen Hinweise einzublenden. Danach aktivieren Sie im Debug-Bereich die Anzeige der gewünschten Hinweise. Wenn Sie nach einem Refresh das Frontend betrachten, sehen Sie Pfad und Dateinamen für alle Template-Vorlagen sowie die eingesetzten Blocknamen, z.B. `/page/2columns-right.phtml` und Mage_Page_Block_Html. So wissen Sie genau, welche Template-Dateien Sie bearbeiten müssen.

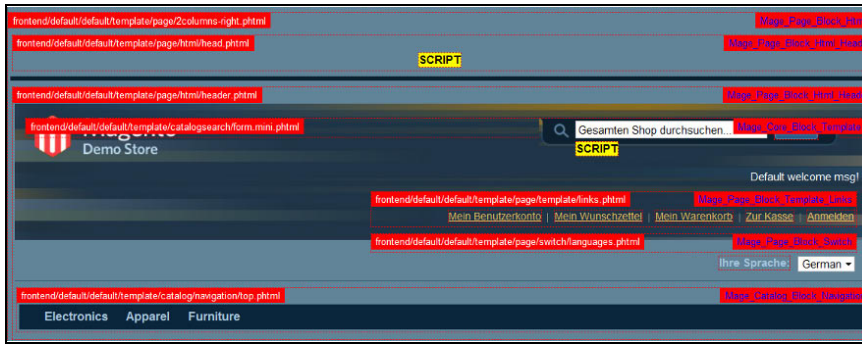

Abbildung 3.10: Vorlagen Pfadhinweise und Blocknamen einblenden

Tipp

Mozilla Firefox-Add-on Firebug

Die Anzeige des Mozilla Firefox-Add-ons Firebug ähnelt etwas der Anzeige der Designhilfe von Magento. Mit diesem Entwicklungswerkzeug steht Ihnen direkt im Webbrowser ein sehr nützliches Tool zur Verfügung. Damit editieren, debuggen oder untersuchen Sie live in der aktuell im Browser angezeigten Webseite CSS-, HTML- und JavaScript-Code. Mittels eines Klicks auf den Button »Untersuchen« identifizieren Sie ganz bequem die verwendeten Container, Klassen oder IDs für die ***CSS****-Bearbeitung. In Abbildung 3.11 sehen Sie genau, welcher Code für die Formatierung der* ***Breadcrumb*** *verantwortlich ist. Gleichzeitig bekommen Sie in der rechten Spalte mitgeteilt, in welcher Datei sich der CSS-Code (z.B. boxes.css aus* `/skin/frontend/default/default/css`*) befindet.*

Übrigens erreichen Sie im aktuellen Microsoft Internet Explorer 8 über die Taste F12 *oder alternativ über »Extras › Entwicklertools« eine Konsole, die ähnlich wie Firebug funktioniert: HTML, JavaScript, CSS und einige Debugging Tools.*

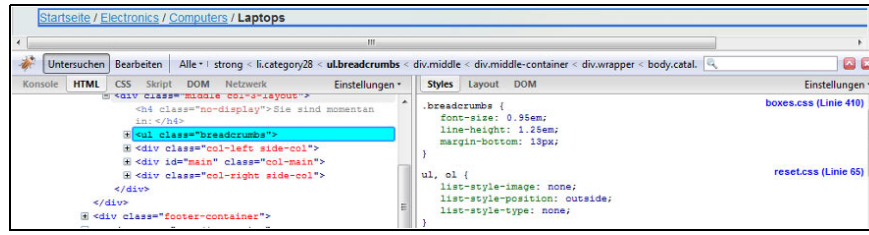

Abbildung 3.11: Firebug zeigt CSS-Quellcode für Breadcrumb-Formatierung

> *magentocommerce.com/wiki/general/opening_phtml_files_in_dreamweaver*
> MagentoCommerce Wiki (Opening PHTML Files in Dreamweaver)

Sortieren der Blöcke

Nachdem Sie einige optische Änderungen und **Textauszeichnungen** per HTML und CSS vorgenommen haben, möchten Sie diese sicherlich genauer begutachten. Den letzten Feinschliff und die **Sortierung** der Blöcke nehmen Sie zuvor noch in den **Layoutdateien** vor. Wie die Template-Dateien, speichert das Shop-System die Layoutdateien auf Modulbasis ab. Mit den Pfadhinweisen zur Template-Datei bekommen Sie daher ebenfalls die Pfadangabe zur Layoutdatei.

Template-Dateien finden

Möchten Sie beispielsweise den **Mini-Warenkorb** im Layout verschieben, dann betrachten Sie zuerst den Template-Pfadhinweis `frontend/default/default/template/checkout/cart/sidebar.phtml`. Darin finden Sie die erforderliche Information. Suchen Sie nach dem ersten Ordnernamen, der nach dem Verzeichnis `/template` folgt. Das ist der Name des Moduls, in unserem Beispiel lautet der **Modulname** checkout. Diesen Namen benötigen Sie, um die zugehörige Layoutdatei zu finden. Die Datei **catalog.xml** ist die Layoutdatei für das catalog-Modul (Katalog), während die Datei **customer.xml** zuständig ist für das customer-Modul (Kunden). Gehen Sie daher in den Layoutordner Ihres Themas, öffnen Sie die Datei **checkout.xml** und verändern Sie die Position des Warenkorbs. Jeder Bereich des Seitenlayouts ist deutlich mit **Kommentaren** gekennzeichnet, die Sie an den öffnenden und schließenden Klammern erkennen <!-- … -->.

Aufbau des Layouts besser verstehen und ändern

Im oberen Abschnitt dieses Kapitels haben wir Ihnen schon einige Hilfsmittel gezeigt, mit denen Sie die relevanten Stellen in den Dateien identifizieren: Templates, Blöcke, CSS-Layout, PHP-Quellcode. Am Ende

dieses Kapitels beschreiben wir Ihnen noch ein paar Tools, die Sie für die Bearbeitung von Dateien einsetzen können. Es handelt sich dabei zwar um Profi-Entwicklungswerkzeuge, aber selbst ein Programmiereinsteiger kommt damit auf Dauer besser zurecht. Jetzt widmen wir uns dem Layoutaufbau und den damit verbundenen Dateien.

Probleme vermeiden

Das Layout ist die virtuelle Komponente der Shop-Applikation. Das Template und die darin enthaltenen Layoutdateien sind für das Aussehen Ihres Shop-Frontends zuständig. Ändern Sie nichts am Core-Quellcode, sonst kann es möglicherweise zu Problemen kommen, wenn diese Dateien durch ein Update verändert werden. Die Trennung von Quellcode und Layout ist ein wichtiges Grundprinzip für die saubere Entwicklungsarbeit mit Webapplikationen. Sie sind in der Lage, über das Template alle gewünschten Layoutänderungen flexibel vorzunehmen, und brauchen dennoch keine Angst zu haben, denn der Quellcode-**Kern** (Core) des *Magento*-Systems bleibt davon unberührt. Die Layoutdateien bestehen aus unkomplizierten XML-Tags, die als Layoutkommandos dienen. Das dringt nicht sehr tief in die Programmierung ein und ist mit etwas Eigeninitiative leicht erlernbar. Mit den einzelnen XML-Tags können Sie die Beziehung zwischen den strukturellen Content-Blöcken modifizieren oder zuweisen. Damit steuern Sie die Frontend-Funktionalität durch Laden oder nicht Laden blockspezifischer Inhalte auf der Seite.

Aufbau Layout-Datei

Eine Layoutdatei besteht aus einer kleinen Anzahl an XML-Tags, die dem Shop-System als genaue Anweisung dienen, wie eine Seite aufgebaut werden soll. Der einfachste Weg, sich damit besser zurechtzufinden, ist Learning-by-doing, also einfach ausprobieren. Die wichtigsten Dinge, die Sie dazu näher betrachten müssen, sind **XML-Handles**, **reference**-Verweise sowie strukturelle und inhaltliche **Blöcke**.

Die Shop-Anwendung selbst trennt in der Layoutausgabe die einzelnen Blöcke voneinander mit sogenannten XML-**Handles**. Das ist eine Art Kennung (**Identifier**), mit deren Hilfe die Anwendung entscheidet, was zu tun ist. Diese Handles verwenden üblicherweise die Namenskonvention *modul_controller_action*. In der Datei checkout.xml befinden sich die Handles `<checkout_multishipping_login>` und `<checkout_multishipping_register>`.

Listing 3.5: Auszug mit Handles in der Layoutdatei checkout.xml

```
<checkout_multishipping_login>
  <update handle="customer_account_login"/>
</checkout_multishipping_login>
<checkout_multishipping_register>
  <update handle="customer_account_create"/>
</checkout_multishipping_register>
```

Update-Layout

Die Shop-Applikation unterscheidet zwei Arten von Layouts: das **Default-Layout** und das **Update-Layout**. Das Default-Layout (z.B. **page.xml**) wird nahezu auf jeder Seite im Shop-Store angewendet, mit Ausnahme weniger Spezialseiten (Bilder-Popup). Alle anderen Update-Anweisungen in Layoutdateien gehören zum zweiten **Layouttyp**, der das geladene Default-Layout auf Seitenbasis überschreibt, also quasi wie ein Update darübergestülpt wird.

Das Default-Layout legt ein spezieller Identifier fest, dieser Handle trägt den Namen `<default>`. In diesem Fall werden die eingebauten Anweisungen dieses Handle auf jeder Seite geladen, erst im Anschluss daran kommen die seitenspezifischen Änderungen an die Reihe. Alle Handles mit einem anderen Namen kommen nur auf der Seite zum Einsatz, die für diese Layoutdatei verantwortlich ist. So verändert der Inhalt des **seitenspezifischen** Handle (page-specific layout) `<catalog_product_view>` mit seinen Layout-Updates nur die Seite der **Produktanzeige** (Product View Page), während der Update-Inhalt des Handle `<catalog_product_compare_index>` nur beim **Produktvergleich** (Compare Product page) aktiv ist. An den Namen der verschiedenen Handles brauchen Sie normalerweise nichts zu verändern.

Spalten zuweisen

Als weiteres Beispiel soll wieder die Produktseite dienen. Das normale Layout dieser Seite verwendet die dreispaltige **Skeleton**-Datei 3columns.phtml (Template). Auf der Produktseite ist diese Darstellung jedoch eher ungeeignet und Sie möchten das Layout in eine zweispaltige Darstellung ändern. Dies bewerkstelligen Sie, indem Sie in der Layoutdatei catalog.xml mit der **Action**-Methode **setTemplate** das Skeleton-Template 2columns-right.phtml einbinden. Dieser Vorgang nennt sich im Allgemeinen Update. Mit der Action-Methode **unsetChild** ist es stattdessen möglich, einen Block auszublenden. Dadurch wird nur auf dieser einen Seite der Block entfernt, er bleibt aber auf allen anderen Seiten weiterhin sichtbar. Eine andere Variante bietet in den XML-Dateien die

Befehlszeile <remove name="[Blockname]" />. Noch eine andere Möglichkeit, wie Sie die **Modulausgabe** komplett auf allen Seiten entfernen, finden Sie unter »System › Konfiguration › Erweitert« (System › Configuration › Advanced). Damit schalten Sie unter anderem die Anzeige des **Umfrage**-Moduls (Mage_Poll) oder des **Newsletters** (Mage_Newsletter) ab. Abschließend publizieren Sie alle Änderungen und aktivieren wieder die Cache-Verwaltung im Backend.

Verschiedene Blöcke

Magento bestimmt das Verhalten und die optische Darstellung einer Seite mittels Block-Tags. Wie bereits beschrieben, gibt es die beiden Typen strukturelle Blöcke und Content-Blöcke. Über die zugewiesenen Tag-Attribute lassen sich die Blöcke einfach unterscheiden. Ein struktureller Block beinhaltet für gewöhnlich immer das **as**-Attribut. Im default-Layout finden Sie jede Menge solcher Attribute. Der Grund dafür ist relativ einleuchtend: Das default-Layout bildet die Basis, auf der dann die einzelnen seitenspezifischen Layouts aufsetzen. Im Standardlayout befinden sich beispielsweise die strukturellen Blöcke left, right, content und footer. Weitere Blockattribute sind:

- **type**: Identifier der Modul-Klasse, die die Funktion des Blocks bestimmt
- **name**: Für die Block-Referenzierung erforderlicher Name
- **before / after**: Zwei Wege, die die Position eines Content-Blocks festlegen
- **template**: Die Funktionalität des Blocks wird durch das Template bestimmt
- **action**: Methoden kontrollieren die mannigfache Store-Front-Funktionalität

Einen reference-Verweis benutzen Sie, um sich auf einen anderen Block zu beziehen. Indem Sie eine Referenz zu einem anderen Block herstellen, werden alle Updates innerhalb der reference auf den entsprechenden Block angewendet. Mittels des name-Attributs beziehen Sie sich per reference-Verweis auf einen anderen Block. Die Verbindung erfolgt über den Namen. Erstellen Sie den Verweis mit <reference name="right">, dann zielen Sie auf den Block <block name="right">.