Gino Cremer / Adrian Lambertz

WordPress-Themes entwickeln

HTML5, CSS3, JavaScript und PHP: Praxiswissen und Quellcodes zum Entwurf von WordPress-Themes

FRANZIS

Gino Cremer / Adrian Lambertz

WordPress-Themes entwickeln

Gino Cremer / Adrian Lambertz

WordPress-Themes entwickeln

HTML5, CSS3, JavaScript und PHP: Praxiswissen und Quellcodes zum Entwurf von WordPress-Themes

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.

Alle Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, dass sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung etwaiger Fehler sind Verlag und Autor jederzeit dankbar. Internetadressen oder Versionsnummern stellen den bei Redaktionsschluss verfügbaren Informationsstand dar. Verlag und Autor übernehmen keinerlei Verantwortung oder Haftung für Veränderungen, die sich aus nicht von ihnen zu vertretenden Umständen ergeben. Evtl. beigefügte oder zum Download angebotene Dateien und Informationen dienen ausschließlich der nicht gewerblichen Nutzung. Eine gewerbliche Nutzung ist nur mit Zustimmung des Lizenzinhabers möglich.

© 2013 Franzis Verlag GmbH, 85540 Haar bei München

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträgern oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt.

Die meisten Produktbezeichnungen von Hard- und Software sowie Firmennamen und Firmenlogos, die in diesem Werk genannt werden, sind in der Regel gleichzeitig auch eingetragene Warenzeichen und sollten als solche betrachtet werden. Der Verlag folgt bei den Produktbezeichnungen im Wesentlichen den Schreibweisen der Hersteller.

Programmleitung: Markus Stäuble
Lektorat: Lothar Schlömer
Satz: DTP-Satz A. Kugge, München
art & design: www.ideehoch2.de
Druck: C.H. Beck, Nördlingen
Printed in Germany

ISBN 978-3-645-60230-3

Inhaltsverzeichnis

Bequem und schnell: Alle Code-Beispiele online beziehen!

Sparen Sie sich die Mühe: Alle in diesem Buch vorgestellten Code-Beispiele können Sie bequem per »Copy & Paste« online beziehen unter *www.wordpress-praxis.de/themes*. Das erspart mühevolles Abschreiben aus dem Buch und verhindert zudem Flüchtigkeitsfehler.

2 Theme-Struktur und Aufbau

Gute Themes in WordPress sind keineswegs auf die rein optische Gestaltung begrenzt. Auch wenn zugegebenermaßen die Optik als Erstes ins Auge springt. Ein gutes Theme ist aber in vielen Fällen ein leistungsfähiges Paket, das eine umfassende Funktionalität beinhaltet und zur Verfügung stellt.

Ein Theme besteht also nicht nur aus dem, was Sie an der Oberfläche sehen. Themes bestehen aus vielen Zeilen Code mit leistungsfähigen Funktionen und oftmals jeder Menge CSS und erweitern Ihre WordPress-Basis teilweise beträchtlich. Kostenlose Themes sehen zwar manchmal ganz nett aus, die meisten bieten aber nur eine minimale Funktionalität, für die essentiellen WordPress-Aufgaben.

Da Sie dieses Buch in Händen halten, ist davon auszugehen, dass Sie sich kein 08/15-Theme von der Stange wünschen, sondern selbst Hand anlegen möchten, um Ihre Website mit einem eigens programmierten Theme zu erstellen. Wenn Sie nun also die Entwickler-Neugier gepackt hat, sollten Sie sich die Theme-Maschinerie von WordPress etwas genauer anschauen. Woraus besteht ein Theme genau? Welche Dateien müssen vorhanden sein und wie sollten diese aufgebaut sein? Auf all diese Fragen erhalten Sie in diesem Kapitel Antworten.

Am klarsten wird die WordPress-eigene Theme-Struktur, wenn Sie in einem ersten Schritt das Standard-Theme von WordPress namens *Twenty Thirteen* etwas genauer unter die Lupe nehmen und grundlegend bearbeiten. Twenty Thirteen wird mit jeder neuen WordPress-Installation ausgeliefert.

Bild 2.1: Das Standard-Theme Twenty Thirteen.

2.1 Die Basis: Der Theme-Ordner

Jedes Theme besitzt in WordPress einen eigenen Ordner im Dateisystem, der über FTP recht einfach angesteuert werden kann. Im Falle des Themes Twenty Thirteen nennt sich dieser – erraten – *twentythirteen* und befindet sich im Ordner *wpcontent/themes*. Diese Ordner werden von WordPress einzeln ausgelesen und in der WordPress-Administrationsoberfläche unter »Themes« als eigenständige Themes entsprechend aufgelistet und dargestellt. Ein neuer Ordner ist daher die Basis und der Anfang eines jeden neuen Themes in WordPress. Doch schauen Sie sich nun den Inhalt des Ordners *twentythirteen* etwas genauer an.

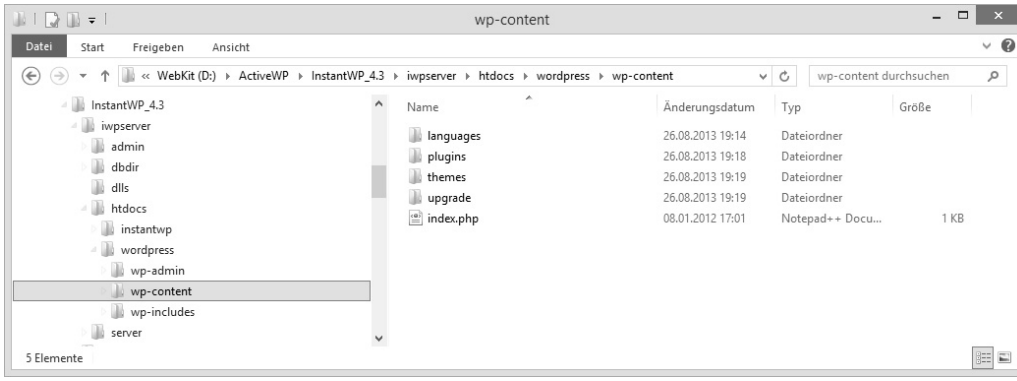

Bild 2.2: Typische Verzeichnisstruktur für eine WordPress-Installation. In diesem Beispiel handelt es sich um eine lokale Installation.

2.2 Die wichtigsten Dateien und Templates

Damit ein Theme als solches von WordPress erkannt wird, müssen neben einem eigenen Unterordner nur zwei Dateien vorhanden sein: Eine *index.php* und eine *style.css*. In der Praxis sind natürlich noch eine Handvoll weiterer Dateien sinnvoll, um wirklich produktiv arbeiten zu können. Aber rein technisch begnügt sich WordPress mit diesen zwei Dateien, um ein Theme als solches in Ihrer WordPress-Administrationsoberfläche zur Aktivierung freizugeben.

Das Theme Twenty Thirteen besteht schon aus sehr vielen Dateien und übertrifft die Mindestanforderungen bei weitem. Logisch, schließlich gilt das Standard-Theme ab Werk als sehr flexibel. Eine Reduktion auf das strikte Minimum würde den Anforderungen sicher nicht gerecht.

Eine einfache HTML-Vorlage, wie sie bei vielen kleineren Websites Anwendung findet, lässt sich mit vier grundlegenden Dateien umsetzen (alle Dateien werden in einem späteren Abschnitt noch genauer durchleuchtet):

- *header.php*: Stellt den Kopfbereich der Website
- *index.php*: Bildet die Hauptseite und den Inhaltsbereich
- *footer.php*: Stellt den Fußbereich der Website
- *style.css*: Stellt alle CSS-Anweisungen zur Verfügung

Diese vier Dateien – und zugegebenermaßen noch einige mehr – finden Sie auch im Ordner *themes/twentythirteen*. Die Datei *index.php* kann als die wichtigste Datei angesehen werden. Wenn Sie schon Websites in HTML umgesetzt haben, wissen Sie vielleicht, dass eine *index.php* (oder im Falle statischer Websites eine *index.html*) immer eine besondere Rolle spielt. Eine Index-Datei wird als Erstes aufgerufen und bildet die

Hauptseite. Alle anderen Dateien, wie die *header.php* und die *footer.php*, werden im Falle von WordPress in diese Datei zusätzlich eingebunden. Im Laufe der Lektüre werden Sie noch zahlreiche andere Dateien kennenlernen, die hinzugefügt werden. So gesehen kann die *index.php* also mit einem Container verglichen werden, während Dateien wie die *header.php* und die *footer.php* wie Pakete hinzugefügt werden.

2.2.1 Aufbau und Struktur der index.php

Öffnen Sie mit einem Editor Ihrer Wahl über eine FTP-Verbindung die Datei *index.php*. Sie finden diese Datei – wie alle anderen Dateien auch – im Ordner *wp-content/themes/twentythirteen/index.php*.

```
<?php
/**
 * The main template file.
 *
 * This is the most generic template file in a WordPress theme and one of the
 * two required files for a theme (the other being style.css).
 * It is used to display a page when nothing more specific matches a query.
 * For example, it puts together the home page when no home.php file exists.
 *
 * Learn more: http://codex.wordpress.org/Template_Hierarchy
 *
 * @package WordPress
 * @subpackage Twenty_Thirteen
 * @since Twenty Thirteen 1.0
 */

get_header(); ?>

	<div id="primary" class="content-area">
		<div id="content" class="site-content" role="main">
		<?php if ( have_posts() ) : ?>

			<?php /* The loop */ ?>
			<?php while ( have_posts() ) : the_post(); ?>
				<?php get_template_part( 'content', get_post_format() ); ?>
			<?php endwhile; ?>

			<?php twentythirteen_paging_nav(); ?>

		<?php else : ?>
			<?php get_template_part( 'content', 'none' ); ?>
		<?php endif; ?>

		</div><!-- #content -->
	</div><!-- #primary -->

<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

Bild 2.3: Der gesamte Inhalt der Datei *index.php* im Überblick. Umrandet und im Fokus: WordPress-Funktionen, die alle Inhalte zusammenstellen.

Wenn Sie die *index.php* des Themes Twenty Thirteen öffnen, werden Ihnen manche Anweisungen sicher schon bekannt vorkommen. Erinnern Sie sich an den PHP-Mini-Crash-Kurs in Kapitel 1.3 ab Seite 17? *get_header()* ist eine dieser typischen WordPress-

Funktionen. Übersetzen könnte man die Bezeichnung *get_header* mit »hole dir den Header«. Entsprechend integriert diese Funktion auf praktische Art und Weise sämtliche Inhalte der Datei *header.php* in diese *index.php*.

Gleichermaßen funktioniert WordPress übrigens mit den beiden Funktionen am Ende des Dokuments: *get_footer();* sorgt dafür, dass alle Inhalte aus der Datei *footer.php* integriert werden, während *get_sidebar();* die Inhalte der Datei *sidebar.php* einbindet, um das notwendige HTML für eine Seitenleiste in Ihre *index.php*-Datei zu laden.

Sie benötigen keine Seitenleiste? Prima, in diesem Fall entfernen Sie einfach die Anweisung <?php get_sidebar(); ?>. Dann wird der Inhalt der *sidebar.php* nicht geladen und eingebunden. Einfach, nicht wahr?

2.2.2 Die wichtigsten Template-Dateien im Überblick

Neben dieser Index-Datei gibt es im Hauptordner des Themes noch jede Menge weitere Template-Dateien, die darauf warten, von Ihnen entdeckt zu werden. Die folgende Auflistung gibt einen Überblick über die Funktionen der einzelnen Dateien.

header.php: Der Kopf der Website

In der Datei *header.php* befinden sich der Code des <head>-Bereichs Ihrer Website sowie der Beginn des <body>-Bereichs. Wenn Sie ein großes Bild im Kopfbereich Ihrer Website darstellen wollen, das auf jeder Seite angezeigt werden soll, können Sie es in die *header.php* setzen. Die Datei *header.php* wird – falls vorhanden – auf jeder Seite eingebunden und aufgerufen. Dadurch ist sie der ideale Ort für Informationen, die auf jeder Seite vorhanden sein sollen.

Gut zu wissen: Head- und Bodybereich einer Website
Jede HTML-Datei besteht aus einem Head- und einem Bodybereich. Während Inhalte im Headbereich nicht direkt sichtbar an die Besucher einer Website ausgeliefert werden, beinhaltet der »Dokumentkörper«, sprich: der Bodybereich, alle Inhalte, die der Besucher der Seite tatsächlich sehen soll. Betrachtet Ihr Seitenbesucher den »nackten« Quelltext, wird er natürlich auch den Head-Bereich erspähen, die Website-Ansicht als solche baut sich aber nur aufgrund des HTML-Konstruktes im Bodybereich auf. Der Head ist optimal geeignet für Meta-Informationen und Javascript- sowie CSS-Verlinkungen.

```
<html>
<head>
<title>Titel der Seite</title>
</head>
<body>
```

```
     Dieser Text wird dem Besucher ausgegeben.
</body>
</html>
```

footer.php: Der Abschluss einer Website

Auch die Datei *footer.php* wird auf jeder Seite aufgerufen und eingebunden. Dadurch ist sie nicht nur geeignet, um Ihrer gesamten Website eine visuelle Fußleiste für z. B. Copyright-Informationen hinzuzufügen. Sie bietet den idealen Abschluss, um alle HTML-Elemente zu schließen, die Sie in der *header.php* geöffnet haben. Sie haben Ihren Body-Bereich mit <body> in der *header.php* geöffnet? Dann müssen Sie den Bereich wieder schließen. Am besten in dieser *footer.php*, da diese auf jeder Seite am Ende aufgerufen wird.

Der Hamburger-Vergleich
Stellen Sie sich die beiden Dateien *header.php* und *footer.php* wie die zwei Hälften eines Hamburger-Brötchens vor. Die obere Hälfte bildet der Header, die untere der Footer. Das sind die Konstanten. Dazwischen findet man je nach Template die unterschiedlichsten »Inhaltszutaten«. Mal eine Kategorienauflistung, mal ein Beitragsdetail. Wie auch immer. Header und Footer bilden immer den Anfang und das Ende Ihrer Website.

page.php: Die Einzelansicht einer Seite

Diese Datei wird aufgerufen, sobald ein Besucher den Inhalt einer statischen Seite anfordert. WordPress unterscheidet in der Administrationsoberfläche zwischen Beiträgen und Seiten. Die *page.php* ist Ihre Anlaufstelle, wenn Sie die Ausgabe einer Seite beinflussen möchten.

single.php: Die Einzelansicht eines Beitrags

Wenn Sie indes steuern möchten, wie der Inhalt eines Beitrags ausgegeben werden soll, sind Sie bei der *single.php* goldrichtig.

Unterschied zwischen Beiträgen und Seiten
Die Beiträge bilden für gewöhnlich das Herz eines WordPress-Auftritts. In ihnen erscheinen die aktuellen Blog-Artikel meistens in umgekehrter chronologischer Reihenfolge, also die neusten zuerst.

Archive, Tags oder Kategorien und weitere Widgets beziehen sich auf die Artikel-Beiträge und nicht auf angelegte Seiten. Sie können z. B. dazu dienen, die Beiträge in ganz unterschiedlicher Zusammensetzung (dynamisch) auszugeben, also zum Beispiel alle Beiträge der Kategorie *Software* oder alle Beiträge vom vorletzten Monat. Auch die Kommentarfunktion für die Leser erscheint unterhalb der Beiträge. Falls ein RSS-Feed für den Blog angelegt wird, gibt er die Beiträge aus, nicht aber die Seiten.
Die Seiten hingegen werden für dauerhafte, statische Inhalte angelegt. Einfache Beispiele sind die Über-uns- und die Impressum-Seite. Der Nutzen der Seite wird aber kaum eingeschränkt, es können auch umfangreiche Workshops oder Bildergalerien dort abgelegt werden. Häufig werden die Seiten in die Hauptnavigation der Website aufgenommen.
Seiten können im Gegensatz zu Beiträgen in WordPress übrigens hierarchisch verschachtelt werden. So kann eine Seite als Elternelement ausgewiesen werden, während andere als Kindelemente dieser Seite untergeordnet werden. So lassen sich mit Seiten und verschachtelten Elementen klassische Website-Strukturen realisieren. Das ist mit Beiträgen so direkt nicht möglich.

sidebar.php: Die Seitenleiste

Die *sidebar.php* ist eine optionale Datei. Sie kann eingesetzt werden, um eine Seitenleiste in Ihr Theme einzubetten. Sie möchten eine Seitenleiste nur in der Einzelansicht eines Beitrags anzeigen lassen, nicht aber auf einer statischen Seite? Rufen Sie über die Funktion *get_sidebar();* die Seitenleiste nur in der Datei *single.php* auf, nicht aber in der Datei *page.php*. Das war's.

functions.php: Raum für Erweiterung

Die Datei *functions.php* kann all Ihre Theme-spezifischen Funktionen beherbergen. Zwar ist die Datei nicht zwingend notwendig, damit ein Theme funktioniert, doch wird kaum ein Theme ohne diese Datei auskommen wollen. Sie werden im späteren Verlauf der Lektüre noch feststellen, wie wichtig diese Datei für die Erweiterung Ihres Themes ist.

Plug-ins vs. Themes
Während Plug-ins bei WordPress einen allgemeinen Einfluss auf das gesamte System nehmen (unabhängig vom genutzten Theme), können Sie Ihr Theme mit Funktionen anreichern, die nur wirken, wenn Ihr Theme tatsächlich genutzt wird. So können Sie – vornehmlich dank der vorhin erwähnten *functions.php* – Ihr Theme um Funktionen erweitern, die sonst nur Plug-ins vorbehalten wären.

An dieser Stelle sollte jedoch beachtet werden, dass ein Theme in erster Linie ein Theme bleiben soll und der Anwender mittels Plug-ins selbst entscheiden sollte, welche Funktionen er wünscht und welche nicht. Viele kommerzielle Themes wirken in dieser Hinsicht funktionsüberladen und sind entsprechend langsam. Halten Sie Ihr Theme so schlank wie möglich.

category.php: Auflistung einer Kategorie

Diese Datei wird für Kategorieauflistungen genutzt und aufgerufen. Ruft ein Besucher eine Kategorie auf, möchte er in der Regel alle Beiträge einsehen, die dieser Kategorie zugeordnet sind. Mit dieser Datei nehmen Sie Einfluss auf die Ausgabe dieser Auflistung.

tag.php: Die Schlagwörter-Auflistung

Identisch mit der *category.php*, nur auf Tags gemünzt.

Was sind Kategorien und Tags?
Im Falle von Blogs sind Kategorien und Schlagwörter (Tags, wie sie oftmals genannt werden) eine absolute Pflicht. Kategorien und Tags erlauben es, Beiträge mit wichtigen Zusatzinformationen zu spicken, zu ordnen, zu klassifizieren. Während Kategorien dazu dienen, einen Beitrag grob einzuordnen, zielen Tags eher darauf ab, den Inhalt selbst zu beschreiben.

archive.php: Das Beitragsarchiv

Typisch für Blogs ist die Ausgabe der Artikel in chronologischer Reihenfolge. Blogs besitzen hierzu passend nach Monaten und Jahren gestaffelte Beitragsarchive. Auf diese Form der Auflistung kann mittels der *archive.php* Einfluss genommen werden.

author.php: Beiträge eines Autors auflisten

Ähnlich wie die *category.php* gibt diese Datei auch eine Liste an Beiträgen aus. In diesem Fall jedoch nicht die Beiträge einer gewissen Kategorie, sondern alle Beiträge, die einem Autor zugewiesen sind.

search.php: Die Suchergebnisse

Die Datei *search.php* kümmert sich um die Ausgabe der Suchresultate. Ihre Website besitzt eine Suchfunktion und Sie möchten Einfluss auf die Ausgabe der Suchresultate nehmen? Dann sind Sie hier richtig.

404.php: Die Fehlerseite

Diese Datei steuert die Ausgabe der 404-Fehlermeldung. Diese Fehlermeldung wird einem Besucher angezeigt, wenn eine Seite oder ein Beitrag nicht gefunden werden konnte.

Warum eigentlich 404?
404 ist ein klassischer Statuscode im Netz, der durch einen Webserver ausgegeben wird, wenn eine Seite nicht gefunden werden konnte. Im direkten Zusammenhang spricht man auch von toten Links, die eine 404-Meldung unmittelbar zur Folge haben. Webserver kennen eine ganze Palette an solchen Statuscodes. Nicht alle bezeichnen Fehler. Der Statuscode 404 ist sicherlich einer der geläufigsten, weil direkt für den Besucher einer Website sichtbar. Vielleicht wurden Sie auch mal mit dem Fehlercode 500 konfrontiert. In diesem Fall sendete der Server selbst S.O.S. (»Internal Server Error«). Mehr zu den verschiedenen Fehlercodes erfahren Sie im Netz[3].

comments.php: Die Kommentarfunktion

Steuert die Ausgabe der User-Kommentare. Diese Datei wird meist innerhalb der Datei *single.php* aufgerufen, sodass die User-Kommentare entsprechend unterhalb eines Beitrags dargestellt werden. Neben der Ausgabe der Kommentare wird in der Regel ein entsprechendes Formular mit ausgegeben, damit Seitenbesucher neue Kommentare einreichen oder auf bestehende antworten können.

2.3 Die Template-Struktur von WordPress

Wie Sie anhand der Auflistung erkennen können, bietet WordPress für jeden Anwendungszweck das passende Template. Wenn Sie Einfluss auf eine statische Seite nehmen wollen, ändern Sie die *page.php*. Möchten Sie Einfluss auf die Suchergebnis-Seite nehmen, ändern Sie die *search.php*. So weit, so gut. Für allgemeingültige Anwendungsszenarien ist damit hinreichend gesorgt. Doch in der Praxis werden Sie des Öfteren individuelle Vorlagen nutzen müssen, um ganz gezielt reagieren zu können.

2.3.1 Unterschiedliche Templates für jeden Anwendungszweck

Einmal angenommen, Sie betreiben ein Nachrichtenportal für Bürger. Sie möchten Ihren Lesern unterschiedliche Ressorts anbieten. So unterteilen Sie Ihre Beiträge in verschiedene Kategorien: Sport, Lokales, Regionales, Nationales.

[3] http://de.wikipedia.org/wiki/Fehlerseite

Wären alle Kategorieauflistungen identisch, könnten Sie bequem die Template-Datei ändern: *category.php*. Diese ist, wie vorhin angeschnitten, allgemein zuständig für Kategorieauflistungen. Doch WordPress bietet Ihnen die Möglichkeit, individuelle Templates und Vorlagen zu erstellen, die genau dort zum Einsatz kommen, wo Sie es sich wünschen. Im Falle des Nachrichtenportals könnten Sie so eigene Vorlagen nutzen für die verschiedenen Ressorts. WordPress setzt dazu eine ganz bestimmte Technik ein und folgt einem klaren Muster, um herauszufinden, welches Template wann zu greifen hat.

2.3.2 ID und Titelform als Auswahlkriterium im Dateinamen

Der Dateiname allein ist für WordPress bereits ein wichtiger Schlüssel, um erkennen zu können, wo ein Template konkret eingesetzt werden soll.

Darüber hinaus besitzt jede Kategorie, jeder Tag, jeder Beitrag, jede Seite, ja sogar jeder Autor in WordPress eine feste ID sowie eine sogenannte Titelform.

Die Dateinamen der Templates stellen sich immer nach dem Muster »*basistemplate-titelform.php*« oder »*basistemplate-ID.php*« zusammen.

Ein paar Beispiele gängiger Templates: *category-13.php*, *page-kontakt.php*, *author-2.php*, *tag-begriff.php* ...

Auch Seiten und Beiträge haben jeweils eigene IDs. Möchten Sie also für eine Seite mit der ID 48 ein eigenes Template bereitstellen, benennen Sie die Datei *page-48.php*.

Wo finde ich die ID und die Titelform?
Rufen Sie in Ihrer WordPress-Administrationsoberfläche die Kategorien auf, finden Sie in der Spalte »Titelform« die notwendige Bezeichnung.
Alternativ können Sie auch IDs nutzen. Diese IDs sind in der Standardeinstellung etwas verborgen. Um beispielsweise die ID einer Kategorie herauszufinden, führen Sie in Ihrer Administrationsoberfläche unter *Beiträge* > *Kategorien* die Maus über den Namen einer Kategorie und betrachten den entsprechenden Link. In dieser Link-Adresse (URL) steht irgendwo *tag_ID=*, gefolgt von einer Zahl. Das gleiche Prinzip können Sie auch für Seiten oder Beiträge anwenden. Diese Zahl ist die ID. Bedeutend praktischer ist aber der Einsatz des Plug-ins *Reveal IDs*[4]. Dieses Plug-in lässt eine neue Spalte »ID« erscheinen.

[4] http://wordpress.org/plugins/reveal-ids-for-wp-admin-25/

Bild 2.4: Praktisches Helferlein: Dank des Plug-ins *Reveal IDs* wird die ID direkt in der Kategorieauflistung ausgegeben. Die Titelform ist auch ohne Plug-in unmittelbar zu erkennen.

Verwechslungsgefahr: Seiten und Beiträge
Sollte ein Template nicht wie erhofft greifen und funktionieren, stellen Sie sicher, dass Sie den Dateinamen richtig gewählt haben. Es kann z. B. durchaus passieren, dass Sie versuchen, einen Beitrag mit der ID 68 über ein Template *page-68.php* anzusprechen. Dabei wäre an dieser Stelle *single-68.php* (single für Beitrag) die richtige Wahl. Gleichermaßen verhält es sich z. B. bei einer statischen Seite mit der Titelform *kontakt*, welche Sie vielleicht irrtümlicherweise versuchen, mit dem Template *single-kontakt.php* anzusprechen. An dieser Stelle müssen Sie für *page-kontakt.php* optieren.

In der Praxis können Sie so Duplikate der allgemeingültigen Templates *category.php* oder *page.php* erstellen, die Dateinamen mit der jeweiligen ID ausstatten und Anpassungen anbringen.

2.3.3 Reihenfolge der Template-Kaskade

Beim Aufruf einer Seite folgt WordPress einer klaren Reihenfolge, was die Kaskadierung der Templates anbelangt. Die Hierarchie gestaltet sich folgendermaßen:

1. Ist ein Template mit einer Titelform vorhanden, z. B. *category-news.php*?
2. Wenn nein, ist ein Template mit einer ID vorhanden, z. B. *category-16.php*?
3. Wenn nein, ist ein Template *category.php* vorhanden?
4. Wenn nein, ist ein Template *archive.php* vorhanden?
5. Wenn nein, greift WordPress in letzter Instanz auf die *index.php*-Datei zurück.

Anhand dieser Vorgehensweise können Sie sich also denken, welchen Wert die Datei *index.php* hat. In vielen Fällen ist sie der letzte Ankerpunkt und kommt zum Tragen, sobald kein spezifischeres Template gefunden werden konnte.

2.4 CSS-Einbindung in Themes

Alle bis hierhin vorgestellten PHP-Dateien dienen der Ausgabe und der Struktur der Website. Aufbau und Optik einer jeden modernen Website werden über eine CSS-Datei gesteuert, die alle relevanten CSS-Anweisungen beinhaltet. In WordPress heißt die zentrale CSS-Datei *style.css*. Alle CSS-Anweisungen können direkt in diese Datei gesetzt werden.

2.5 Welchen Zweck erfüllen die CSS-Kommentare zu Beginn?

Wenn Sie diese CSS-Datei im Falle des WordPress-Standard-Themes Twenty Thirteen öffnen, stellen Sie fest, dass ganz oben jede Menge CSS-Kommentare eingefügt wurden. Lassen Sie sich nicht von der Menge irritieren.

```
/*
Theme Name: Twenty Thirteen
Theme URI: http://wordpress.org/extend/themes/twentythirteen
Author: the WordPress team
Author URI: http://wordpress.org/
Description: The 2013 theme for WordPress takes us back to the blog, featuring a full range of
post formats, each displayed beautifully in their own unique way. Design details abound, starting
with a gorgeous color scheme and matching header images, optional display fonts for beautiful
typography, and a wide layout that looks great on large screens yet remains device-agnostic and
is readable on any device.
Version: 0.1
License: GNU General Public License v2 or later
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Tags: black, brown, orange, tan, white, yellow, light, one-column, two-columns, right-sidebar,
flexible-width, custom-header, custom-menu, editor-style, featured-images, microformats, post-
formats, rtl-language-support, sticky-post, translation-ready
Text Domain: twentythirteen

This theme, like WordPress, is licensed under the GPL.
Use it to make something cool, have fun, and share what you've learned with others.
*/


/**
 * Table of Contents:
 *
 * 1.0 - Reset
 * 2.0 - Repeatable Patterns
 * 3.0 - Site Structure
 * 4.0 - Header
 *   4.1 - Site Header
 *   4.2 - Navigation
 * 5.0 - Content
```

Bild 2.5: Ausführliche Meta-Information in der *style.css* von Twenty Thirteen.

Das Theme Twenty Thirteen ist natürlich wieder ein Musterbeispiel an Ausführlichkeit. Der obere Bereich bietet nicht nur Platz für wertvolle Notizen und Informationen. Reduziert man diese Kommentare aufs Wesentliche, bleibt nicht mehr so viel übrig und alles wirkt direkt bedeutend übersichtlicher:

```
/*
Theme Name: Twenty Thirteen
Theme URI: http://wordpress.org/extend/themes/twentythirteen
Author: the WordPress team
Author URI: http://wordpress.org/
Description: The 2013 theme for WordPress takes us back to the blog
*/
```

Bild 2.6: Aufgeräumt und übersichtlich: Mehr wäre im Prinzip nicht notwendig.

Wechselt man nun in der WordPress-Administrationsoberfläche in die Rubrik *Design* > *Themes*, wird man haargenau die in der *style.css*-Datei festgelegten Meta-Informationen im Administrationsbereich wiederfinden.

Bild 2.7: Ausgabe der CSS-Kommentare direkt in der Theme-Übersicht im WordPress Administrationsbereich.

Sie erhalten also dank dieser CSS-Kommentare die Möglichkeit, Ihr Theme entsprechend mit Meta-Informationen auszustatten. Wer ist der Autor? Soll der Autor sogar verlinkt sein (Author URI)? Gibt es eine Beschreibung (Description)? Und wie heißt das Theme überhaupt (Theme Name)? Wenn Sie Ihr Theme für sich selbst entwickeln, sind diese Informationen natürlich eher kosmetischer Natur. Sollten Sie Ihr Theme allerdings für einen Kunden entwickeln, der Zugang zur Administrationsoberfläche hat, wirken diese Informationen professionell (zudem können Sie sich als Autor und Urheber des Themes hervorheben). Wenn Sie sogar gedenken, Ihr Theme zu veröffentlichen, sollten Sie diesen Platz nutzen, um z. B. auf eine passende Website zu verlinken, die nähere Informationen zu dem Theme bereithält.

2.6 Der Loop unter der Lupe: Es werde Inhalt!

Nachdem Sie nun an der Oberfläche »schnuppern« durften, sind Sie sicher schon ganz gespannt darauf, die Templates etwas genauer unter die Lupe zu nehmen. Sofern Sie also noch frohen Mutes sind (ein Kaffee zwischendurch wirkt übrigens Wunder), sollten Sie nun einen weiteren Blick in die Datei *index.php* werfen.

Bevor es weitergeht: Kopf hoch!
Bevor Sie mit Technik versorgt werden, ein paar aufmunternde Worte zu Beginn: Ob Sie nun die *index.php* genauer betrachten, die *category.php* oder die *single.php* (oder ein anderes Template Ihrer Wahl): Der Aufbau ist im Prinzip immer gleich. Das ist das Schöne an WordPress-Themes. Ein paar PHP-Basics und eine Handvoll WordPress-Funktionen reichen, und Sie haben bereits einen guten Durchblick.

2.6.1 Allgemein und einfach: Die index.php

Öffnen Sie nun die *index.php* direkt im Hauptordner des Themes Twenty Thirteen. Wenn Sie das Grundlagen-Kapitel aufmerksam gelesen haben, dürften Ihnen die wesentlichen Elemente in dieser Datei zumindest schon mal bekannt vorkommen. Die *index.php* versammelt im Prinzip alle im Grundlagen-Kapitel vorgestellten Eigenschaften.

Unterschiede zwischen Twenty Twelve und Twenty Thirteen
WordPress benennt seine Standard-Themes immer nach den jeweiligen Erscheinungsjahren. Das Vorgänger-Theme des aktuellen Themes Twenty Thirteen nennt sich ... Twenty Twelve. Die Datei *index.php* von Twenty Twelve war noch bedeutend umfangreicher und unübersichtlicher. Im aktuellen Theme Twenty Thirteen werden der Übersicht halber viele Codeteile entsprechend ausgelagert. Wundern Sie sich also nicht, wenn Sie die *index.php* des Themes Twenty Twelve öffnen.

```
<?php
/**
 * The main template file.
 *
 * This is the most generic template file in a WordPress theme and one of the
 * two required files for a theme (the other being style.css).
 * It is used to display a page when nothing more specific matches a query.
 * For example, it puts together the home page when no home.php file exists.
 *
 * Learn more: http://codex.wordpress.org/Template_Hierarchy
 *
 * @package WordPress
 * @subpackage Twenty_Thirteen
 * @since Twenty Thirteen 1.0
 */

get_header(); ?>

	<div id="primary" class="content-area">
		<div id="content" class="site-content" role="main">
		<?php if ( have_posts() ) : ?>

			<?php /* The loop */ ?>
			<?php while ( have_posts() ) : the_post(); ?>
				<?php get_template_part( 'content', get_post_format() ); ?>
			<?php endwhile; ?>

			<?php twentythirteen_paging_nav(); ?>

		<?php else : ?>
			<?php get_template_part( 'content', 'none' ); ?>
		<?php endif; ?>

		</div><!-- #content -->
	</div><!-- #primary -->

<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

Bild 2.8: Viel Bekanntes in der *index.php*: Header, Footer, Loop.

Während vorhin nur der Anfang und das Ende der Datei *index.php* beleuchtet wurden (Sie erinnern sich sicher an den schmackhaften Hamburger), widmen wir uns nun dem spannenden Mittelteil. Dieser ist nämlich zuständig für die Ausgabe der Inhalte und nennt sich »Loop«. Ein Loop in WordPress kann im Rahmen einer Einzelansicht – also beispielsweise bei Aufruf des Templates *single.php* – einen einzelnen Beitrag auslesen oder – im Rahmen einer Auflistung wie der *category.php* – zahlreiche Beiträge gleichzeitig auslesen.

Den Loop starten

Die Auslese-Schleife beginnt bei *while* und endet – erraten – bei *endwhile*. In diesem Fall ist der Loop sehr überschaubar.

```
<?php while ( have_posts() ) : the_post(); ?>
   <?php get_template_part( 'content', get_post_format() ); ?>
<?php endwhile; ?>
```

Solange es Beiträge gibt, sollen diese auch jeweils ausgegeben werden. Diese Aufgabe übernimmt die WordPress-Funktion *get_template_part()*.

Die WordPress-Funktion *get_template_part()* ist sehr mächtig und erlaubt ein Hinzufügen von Template-Teilen aus externen Dateien direkt in den Loop. Der Vorteil: Dank dieser Funktion konnte die *index.php* in Twenty Thirteen entsprechend eingedampft werden. Wurde vorher die *index.php* in Twenty Twelve ziemlich aufgebläht, wurden nun viele Codeteile ausgelagert und mit dieser Funktion einfach reingeholt und aufgerufen.

Das Theme mit einem Screenshot abrunden
Sicherlich haben Sie sich schon gefragt, woher der Bildschirmausschnitt des Themes im Administrationsbereich stammt. Dieses Bild kann einfach als Bilddatei in den Hauptordner gelegt werden und muss den Dateinamen *screenshot.png* haben. Dementsprechend darf es auch kein Foto im JPG-Format sein, sondern muss eine PNG-Datei sein. Die kann man mit einem handelsüblichen Bildbearbeitungsprogramm erstellen.

2.7 Zusammenfassung: So sieht ein Theme aus

Sie haben nun die wichtigsten Templates kennengelernt, wurden mit zahlreichen neuen Begriffen und Dateien konfrontiert und wissen, wie ein Theme von der Basis her aufgebaut ist. Zusammenfassend lässt sich sagen, dass ein Theme in WordPress aus folgenden Elementen besteht:

- Die Datei *index.php* bildet das Rückgrat und ist die wichtigste Theme-Datei. Sie wird auch immer genutzt, sobald keine anderen Templates in Frage kommen.
- Jedes Theme muss eine CSS-Datei mit dem Namen *style.css* haben. Wenn Sie Ihrem Theme einen Titel, eine Beschreibung und weitere Meta-Informationen mitgeben möchten, schreiben Sie diese einfach als CSS-Kommentar in den Kopf der CSS-Datei.
- Wichtige Templates sind die *header.php* und die *footer.php*. Ähnlich wie ein Sandwich oder Hamburger bilden sie sozusagen Kopf und Füße Ihrer Website.
- Templates werden über eigene WordPress-Funktionen einfach eingebunden. Mit *get_sidebar();* wird die Datei *sidebar.php* geladen. Analog können Sie mit *get_header();* und *get_footer(); header.php* und *footer.php* aufrufen und einbinden.
- Sie können beliebig viele eigene Templates erstellen. WordPress selektiert diese nach dem Dateinamen. Gibt es beispielsweise zur Auflistung der Beiträge in der Kategorie »Technik« keine Datei *category-technik.php*, sucht WordPress nach der *category-ID.php*. Ist auch diese nicht vorhanden, greift WordPress einfach auf die für alle Kategorieauflistungen gleichermaßen gültige *category.php* zurück. Ist keine *category.php* da? Kein Problem, WordPress greift als »Ultima Ratio« zur *index.php*.

- Jedes Theme kann mit einer Bilddatei *screenshot.png* ein eigenes Vorschaubild für den Administrationsbereich erhalten.
- Zentraler Bestandteil von vielen Templates ist der Loop. Diese Schleife listet die jeweiligen (gefilterten) Inhalte auf.
- Ein Loop ist eine Schleife, welche in PHP mit *while* begonnen und mit *endwhile* beendet wird.
- Ebenfalls oft vorhanden: *if/else*-Anweisungen. Wenn (*if*) diese oder jene Bedingung erfüllt ist, tue das. Ansonsten (*else*) tue folgendes.

Stichwortverzeichnis

Gino Cremer / Adrian Lambertz

WordPress-Themes entwickeln

Viele Internetseiten werden mit WordPress umgesetzt, wobei man dem System nicht immer sofort ansieht, dass es mit WordPress läuft. Das liegt an den Themes. WordPress-Themes sind austauschbare Layoutvorlagen für WordPress-Seiten. Die Basis eines Themes bilden HTML5, CSS3, JavaScript und PHP: Die richtige Mischung schafft das optimale Theme. Wenn es richtig gut ist, können Sie es sogar auf diversen Marktplätzen zum Verkauf anbieten.

Lernen Sie die Grammatik

Ein WordPress-Theme folgt einem bestimmten Aufbau. Neben der Ordnerstruktur sind noch andere Konventionen wichtig. Sie alle gilt es zu kennen und zu beachten, wenn Sie ein eigenes Theme umsetzen. Dabei hilft Ihnen dieses Buch. Das notwendige Wissen in PHP bekommen Sie hier ebenfalls.

Nicht das Rad neu erfinden

Im Internet gibt es bereits viele vorhandene Themes für WordPress. Bei der Entwicklung eines eigenen Themes müssen Sie deswegen das Rad nicht neu erfinden, sondern können auf einem bereits vorhandenen Theme aufsetzen. Solche Themes werden als Child-Themes bezeichnet. Schritt für Schritt lernen Sie, wie Sie auf Basis eines vorhandenen Themes ein neues aufsetzen und die gewünschten Funktionen darin umsetzen.

Quellcode für den Projektalltag

Viele Elemente in einem Theme sind auf die eine oder andere Art auch in anderen Themes vorhanden. Ein eigenes Kapitel widmet sich der Wiederverwendung von sogenannten Snippets: Eine Facebook- oder Twitter-Integration zum Beispiel hat fast jede Webseite. Auch das Anbringen von Kommentaren gehört heute bei vielen WordPress-Seiten zum Standard. In praktischen Beispielen wird Ihnen der Einsatz solcher Snippets gezeigt. Damit erweitern Sie sehr schnell Ihr eigenes Theme.

Aus dem Inhalt:

Über die Autoren:

Gino Cremer ist Geschäftsführer der auf Weblösungen spezialisierten Agentur Pixelbar aus dem belgischen Eupen. Er hat langjährige Erfahrung mit CMS-basierten Kundenprojekten, vornehmlich auf WordPress-Basis und ist ein Webdesigner der ersten Stunde. Zur Zeit arbeitet Gino Cremer auch als Dozent und Berater am WIFI Wien im Bereich Social Media und Webdesign.

Adrian Lambertz arbeitet als Webentwickler in der von Gino Cremer geführten Webagentur. Während seiner Ausbildung begann er, sich intensiv mit WordPress auseinanderzusetzen und entwickelt heute moderne, kundenspezifische WordPress-Web-sites nach neuesten Webstandards und -techniken.

9 783645 602303

30,– EUR [D] **/ 30,90 EUR** [A]
ISBN 978-3-645-60230-3

Besuchen Sie unsere Website
www.franzis.de

FRANZIS