2 Grundlagen der WordPress-Theme-Erstellung

WordPress hat bei seiner Plattform etwas Geniales gemacht. Der Inhalt ist vollständig von der Funktionalität und vom Design getrennt. WordPress-Themes und Plugins erlauben Ihnen ganz elegant, Inhalte, Aussehen und Funktionalität vollkommen unabhängig voneinander zu verwalten. Diese *Trennung der Belange* (separation of concerns) ist etwas, was Sie immer im Hinterkopf haben sollten, wenn Sie mit WordPress arbeiten; es ist auch mehrfach Thema in diesem Kapitel. Genau genommen bietet Ihnen dieses Kapitel einen Einstieg in die Programmierung für WordPress und dabei ist es wichtig, dass Sie sich an diese Trennung erinnern.

Es gibt Unmengen von Dateien im WordPress-Ordner, die gut organisiert und intuitiv benannt sind. Alles, worum Sie sich kümmern müssen, befindet sich in */wp-content/*. Dort sind der */plugins/*- und der */themes/*-Ordner.

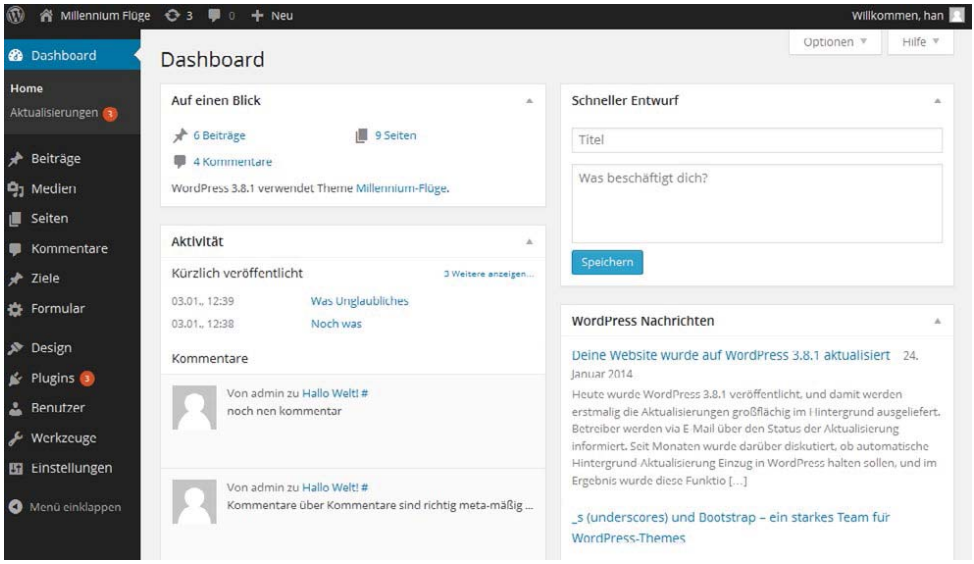

Abb. 2-1 *Das WordPress-Dashboard zeigt Ihnen direkt nach dem Einloggen Ihre letzten Aktivitäten und Statistiken.*

Hinweis

Wenn Sie etwas ändern, das sich nicht im Ordner */wp-content/* befindet, ändern Sie den Kern von WordPress – und das ist eine Todsünde bei der WordPress-Entwicklung. Wenn Sie den Kern von WordPress verändern, führen Sie Änderungen an Dateien durch, die beim nächsten Update überschrieben werden – und solche Updates führen manche Webhoster automatisch durch. Dann sind alle Änderungen verschwunden.

Sie werden die meiste Zeit im */themes/*-Verzeichnis am Erscheinungsbild arbeiten, aber manches, was wir besprechen, gehört eher in die Kategorie Funktionalität oder Inhalt, und funktioniert besser, wenn man es in Plugins auslagert. Schließlich wollen Sie keinen Inhalt und auch keine Funktionalität in das Design Ihrer Webseite einbauen. Um zu entscheiden, ob etwas in ein Theme oder in ein Plugin gehört, sollten Sie sich folgende Frage stellen: »Möchte ich die Funktion behalten, wenn ich die Seite neu designe?« Wenn die Antwort Ja ist (oder Ja sein könnte), sollten Sie daraus ein Plugin machen. Wenn die Antwort hingegen Nein ist, dann gehört das, was Sie hinzufügen wollen, wirklich zum Erscheinungsbild. Und damit gehört es in Ihr Theme.

»Möchte ich die Funktion behalten, wenn ich die Seite neu designe?«

Es gab einen Trend bei kommerziellen WordPress-Themes, mehr Funktionen zu integrieren, damit die Benutzer das Gefühl haben, mehr für ihr Geld zu bekommen. Aber das hat gravierende Nachteile. Wenn Sie ein Theme mit Funktionen anreichern, die nicht direkt zum Design der Website gehören, dann wird der Benutzer die Funktionen bei einem Redesign verlieren. Custom-Post-Types (CPTs = individuelle Inhaltstypen), die es erlauben, verschiedene Arten von Inhalten zu WordPress-Sites hinzuzufügen, sind ein gutes Beispiel für die möglichen Probleme. Nehmen wir an, Sie haben einen Geschäfts-CPT in Ihrem Theme, den Sie nutzen, um ein Firmenverzeichnis zu erstellen. Wenn Sie jetzt Ihr Theme ändern und deaktivieren, *verlieren Sie diese ganzen Inhalte*. Sie müssten die CPTs aus dem Theme kopieren und in das neue einfügen, was für manche Benutzer kein gangbarer Weg ist: Wenn ein Benutzer keinen direkten Zugriff auf den Server hat, kann er das Theme auch nicht auf der Codeebene ändern. Dasselbe betrifft Funktionen wie Bilder-Slider, Zahlungsschnittstellen und alles, was nicht im strengen Sinne zum Design gehört.

Mit dieser Warnung im Kopf können wir uns jetzt genauer mit der Theme-Entwicklung beschäftigen. Wir werden vier Punkte behandeln: die Template-Struktur, den Loop, die Custom-Post-Types und Plugins.

Der WordPress-Codex

Wenn Sie tiefer einsteigen, wird der WordPress-Codex (oder kurz nur »der Codex«) eine unverzichtbare Quelle für Sie werden. Er ist die ausführlichste Dokumentation von WordPress und die erste Anlaufstelle, um etwas über die WordPress-APIs zu erfahren. Dort finden Sie jede Menge Beispielcode, Änderungsprotokolle der einzelnen Versionen und optimale Vorgehensweisen. Kurz gesagt: Der Codex ist Ihr bester Freund bei der Entwicklung von WordPress-Themes und Plugins.

Sie finden den Codex unter http://rwdwp.com/8.

2.1 Gestatten: unsere Website

Nachfolgend sehen Sie die Site, mit der wir das restliche Buch arbeiten werden. Es ist eine Website für ein Weltraum-Reisebüro namens *Millennium Flüge*. In den nächsten Kapiteln werden wir den HTML/CSS-Code in ein vollständig responsives WordPress-Theme verwandeln. Viel Spaß dabei!

Abb. 2-2 *Gestatten: Millennium Flüge, unser zukünftiges WordPress-Theme*

2.2 Die Template-Struktur

Wenn Sie ein Theme entwickeln (die Bezeichnung für ein vollständiges Design einer WordPress-Site), werden Sie einzelne Template-Dateien erstellen, die kontrollieren, wie bestimmte Komponenten der Site dargestellt werden – etwa Beiträge, Seiten, CPTs usw. Unter der Motorhaube von WordPress gibt es eine äußerst differenzierte Template-Hierarchie, die Inhalte in Abhängigkeit davon anzeigt, wie die Theme-Dateien benannt sind. Die einzigen Dateien, die wirklich benötigt werden, damit ein Theme funktioniert, sind *style.css* und *index.php*. Letzteres ist die Datei, die WordPress verwendet, wenn es in einem Theme keine anderen Templates gibt. Gleichzeitig können Sie aber eine Vielzahl von unterschiedlichen Templates für Seiten, einzelne Beiträge, Kategorie-Seiten, Tags, Taxonomien und mehr haben. Für eine noch genauere Kontrolle können Sie auch spezielle Seiten, Tags, Kategorien und Custom-Post-Types definieren. Wenn Sie beispielsweise einen CPT mit Namen »business« haben, so können Sie auch ein Template namens *single-business.php* erstellen, das automatisch verwendet wird, um Beiträge dieser Art anzuzeigen. In dem Fall, dass Sie einen CPT ohne ein spezielles Template haben, wird WordPress stattdessen ein anderes Template verwenden, nämlich *single.php*. Und wenn *single.php* nicht vorhanden ist, *index.php*. Die vollständige Fallback-Struktur sehen Sie in Abb. 2-3, das ist die sogenannte WordPress-Template-Hierarchie.

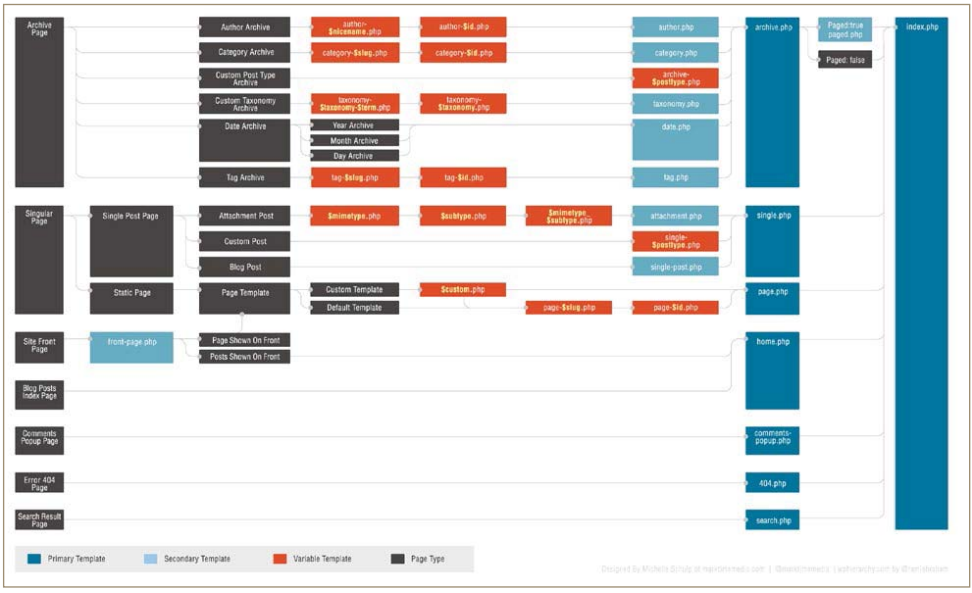

Abb. 2-3 *Die Template-Hierarchie von WordPress. Eine interaktive Version dieses Schaubildes finden Sie unter http://rwdwp.com/9.*

Wir werden uns nicht jedes Detail der Hierarchie ansehen, aber wir beschäftigen uns mit den Grundlagen, bevor wir zu dem spannenderen Teil kommen – der eigentlichen Erstellung eines responsiven Themes.

Tipp

Das sehr gute (und nur auf Englisch erhältliche) Buch *The Web Designer's Guide to WordPress* von Jesse Friedman behandelt viele Aspekte von WordPress, die hier nicht explizit besprochen werden.

2.2.1 style.css

Es ist sinnvoll mit der Datei *style.css* zu beginnen – einfach deswegen, weil dies der Ort ist, wo Ihr Theme definiert wird. Zusätzlich zum sitespezifischen CSS-Code beginnt jede *style.css*-Datei mit einem Codeblock wie dem folgenden:

```
/*
Theme Name: Millennium Flüge
Theme URI: http://www.millenniumflights.com
Description: Ein angepasstes Theme für Millennium Flüge, Inc.
Version: 1.0
Author: Joe Casabona
Author URI: http://www.casabona.org
Tags: blue, white, two-column, flexible-width
License: GPL
License URI: http://www.gnu.org/licenses/gpl.html

Hier können allgemeine Kommentare folgen, sind aber nicht obligatorisch.
*/
```

Dieser Block verrät WordPress alles, was es wissen muss, um das Theme im Administrationsbereich von WordPress unter *Design/Themes* aufzuführen. Das Einzige, was man im Administrationsbereich sieht, das hier nicht im Kommentarblock erwähnt wurde, ist der zugehörige Screenshot. Das ist eine eigene Datei, die entweder *screenshot.png* oder *screenshot.jpg* heißt.

Hinweis

Die hinter `Tags` aufgeführten Begriffe werden von WordPress verwendet, um Ihr Theme aufgrund der aufgeführten Schlüsselbegriffe (Tags) auffindbar zu machen. Die Liste aller erlaubten Tags finden Sie unter http://rwdwp.com/10.

Was den eigentlichen CSS-Code anbelangt, gibt der WordPress-Codex ein paar generelle Empfehlungen, die auf allgemeinen empfohlenen Herangehensweisen basieren; etwa: »verwenden Sie, wenn möglich, validen CSS-Code«, »beschränken Sie browserspezifische Hacks« und »ergänzen Sie eine *print.css*-Datei«. Zusätzlich zu diesen Emp-

fehlungen rate ich Ihnen, den gesamten CSS-Code in *einer* Datei zusammenzufassen. Es ist gängige Praxis, den CSS-Code auf vier verschiedene Dateien aufzuteilen (die Namen können abweichend sein): *reset.css*, *master.css*, *fixes.css* und *styles.css*, wobei die letzte Datei die anderen drei Dateien einbindet. Wie bereits in Kapitel 1 erwähnt, geht es beim RWD um mehr als nur darum, eine Website zu erstellen, die sich in der Größe anpasst. Es geht vielmehr darum, eine Website zu erstellen, die plattformübergreifend eine gute User Experience erlaubt. Wenn ein Benutzer eine beschränkte oder langsame Internetverbindung hat, bedeuten mehr HTTP-Requests eine schlechtere Performance. Anstelle von vier Stylesheets können Sie eines verwenden und damit sparen Sie drei HTTP-Requests, die nur Bandbreite brauchen, die der Nutzer nicht unbedingt hat.

Tipp

Anstelle einer separaten *print.css*-Datei können Sie auch eine Media Query für die Druckformatierungen verwenden. Dafür müssen Sie nur den folgenden Code irgendwo in Ihrem Stylesheet einfügen: `@media print { /* und hier folgt der CSS-Code */ }`.

Eine weitere Empfehlung des Codex ist die Definition einer Reihe von allgemeinen CSS-Styles zur Anordnung von Bildern und zur Gestaltung von Bildunterschriften – was hilfreich ist, wenn Sie gerade beim Designen nicht daran gedacht hatten. Sie finden diese unter http://rwdwp.com/11. Vielleicht noch hilfreicher für beginnende WordPress-Designer ist die vollständige Liste der von WordPress erzeugten CSS-Klassen unter http://rwdwp.com/12.

2.2.2 functions.php

Obwohl ich gesagt habe, dass ein WordPress-Theme nur zwei Dateien benötigt – *styles.css* und *index.php* – möchte ich mir *index.php* für den Schluss aufsparen. Die nächsten drei Dateien, die wir uns ansehen werden, sind wichtig, um eine ordentliche *index.php*-Datei zu erstellen; behandeln wir diese zuerst und beginnen mit *functions.php*.

Die Datei *functions.php* ist der Ort, wo Sie den unterschiedlichsten PHP-Code unterbringen können. Sie können diese Datei verwenden, um Features wie Sidebars, Navigationen, Thumbnails, Ihre eigenen PHP-Funktionen, Theme-Optionen usw. hinzuzufügen. Der Codex beschreibt die *functions.php*-Datei als eine »Plugin«-Datei für Ihr Theme. Die *functions.php*-Datei ist zwar nicht notwendig, aber sie ist äußerst nützlich. Ich benutze meine normalerweise, um Konstanten zu definieren, die ich im Theme brauche. Außerdem integriere ich ein paar allgemeine Funktionen, die ich schon seit Jahren einsetze. Es empfiehlt sich, alles gut zu organisieren, und nicht einfach Kraut und Rüben in diese Datei zu packen, aber prinzipiell kann die *functions.php*-Datei ein zentraler Platz für zusätzliche Funktionalitäten sein.

Hinweis

Wenn Sie mein vorheriges Buch *Building WordPress-Themes from Scratch* gelesen haben, dann erinnern Sie sich vielleicht daran, dass ich gesagt habe, die Datei *functions.php* wäre ebenfalls ein guter Platz für CPTs. Aber – wie bereits am Anfang des Buches erwähnt – empfiehlt es sich heute, aus diesen eigene Plugins zu machen.

In diesem Buch wird *functions.php* benutzt, um die serverseitige Erkennung der Bildschirmauflösung u.Ä. zu bewerkstelligen, sodass Sie verschiedene Templates in Abhängigkeit des Geräts ausliefern können, das der Benutzer verwendet.

Noch ein paar Hinweise zur *functions.php*:

- Sie können WordPress-Hooks einsetzen, um tiefer in WordPress einzugreifen. Sie könnten beispielsweise das RSS-Widget verändern, ein eigenes Logo beim Login-Bildschirm hinzufügen, die Länge des Ausschnitts von Beiträgen bestimmen oder sogar die Ausgabe der `the_content()`-Funktion verändern.
- Sie können Ihre eigenen Funktionen definieren, die auf die gesamte Website angewandt werden, inklusive des Administrationsbereichs. Denken Sie daran, all Ihre Funktionen oder Klassen mit einem Präfix zu versehen, um Konflikte zu vermeiden. Dasselbe sollten Sie auch mit Plugins und Shortcodes machen.
- Sie sollten versuchen, einen Überblick darüber zu behalten, was Sie alles in *functions.php* hinzufügen. Dabei sollten Sie sich immer fragen, was Sie bei einem Redesign behalten möchten. Bei der Arbeit mit dieser Datei können die Dinge leicht etwas unübersichtlich werden.

2.2.3 header.php und footer.php

Die Template-Dateien *header.php* und *footer.php* sind wahrscheinlich diejenigen, die Sie am häufigsten in Ihrem Theme aufrufen, denn sie beinhalten das, was nicht Teil des eigentlichen Inhalts ist. In *header.php* etwa gehört so gut wie alles, was oberhalb des Inhaltsbereichs steht. Dazu zählt `html`, `head` und der `body`-Start-Tag, aber auch beispielsweise der Name der Website, das Suchformular und die Navigation. Um den *header* in ein Template einzubinden, verwenden Sie die `get_header()`-Funktion. Entsprechend steht in der Datei *footer.php* alles, was nach dem Inhaltsbereich kommt. Diese Datei schließt die `body`- und `html`-Tags, und üblicherweise stehen hier auch der Fußbereich der Website, Google Analytics und weitere Dinge, die Sie am Ende vor die `</body>`- und `</html>`-Tags schreiben. Zur Einbindung schreiben Sie `get_footer()`.

Die äußerst wichtige WordPress-Funktion `wp_head()` wird ebenfalls meistens in *header.php* verwendet. Diese Funktion sorgt dafür, dass eine Website alle Skripte, Stylesheets und Informationen von Plugins lädt. Wenn es beispielsweise ein Plugin für einen jQuery-Slider gibt, das `wp_enqueue_script()` nutzt, um JavaScript-Dateien zu laden (oder sogar wenn Sie `wp_enqueue_script()` in Ihrer *functions.php*-Datei schrei-

ben), wird WordPress aufgrund der Funktion `wp_head()` wissen, dass es die JavaScript-Dateien im `head`-Bereich laden muss. Wenn Sie `wp_head()` nicht aufrufen, ist es recht wahrscheinlich, dass die Dinge nicht mehr wie gewünscht funktionieren werden. Sie sollten `wp_head()` direkt vor dem `</head>`-Tag angeben.

Die Datei *footer.php* ist am besten geeignet für das Gegenstück zu `wp_head()`, nämlich `wp_footer()`. Diese Funktion sollte direkt vor dem `</body>`-Tag angegeben werden und sie hat die Aufgabe, alle Skripte, Stylesheets und Texte zu laden, die unten auf der Webseite hinzugefügt werden sollen. Wenn Sie `wp_enqueue_script()` verwenden und den Parameter `$in_footer` auf `true` setzen, wird diese Funktion versuchen, die JavaScript-Datei im Fußbereich zu laden. Ohne `wp_footer()` würde das nicht passieren.

Hinweis

Die einzige Anforderung von `wp_head()` und `wp_footer()` ist, dass diese im `<head>`- bzw. `<body>`-Bereich der Webseite geladen werden müssen, um korrekt zu funktionieren. Wenn Sie eine eigenständige Seite erstellen wollen, die nicht das Theme-Layout benutzt, müssen Sie auch nicht unbedingt `wp_head()` in *header.php* und `wp_footer()` in *footer.php* platzieren, solange sie an der richtigen Stelle stehen.

Abgesehen davon, dass diese Funktionen aufgerufen werden, sehen *header.php* und *footer.php* wie ganz normale Kopf- und Fußbereiche aus, wie Sie sie auf irgendwelchen anderen Webseiten finden. Wo wir gerade dabei sind: Hier sehen Sie die Datei *footer.php*, wie wir sie auf der Millennium-Flüge-Site benutzen werden.

```
</div>
<footer>
  <aside class="group">
    <?php if ( !function_exists( 'dynamic_sidebar' ) || !dynamic_sidebar('Footer') ) : ?>
    //Hier folgen die Widgets des Fußbereichs
    <?php endif; /* (!function_exists('dynamic_sidebar') */ ?>
  </aside>
  <p>&copy; <a href="http://millenniumflights.com">Millennium Flüge</a>, <?php
     print date('Y'); ?>.</p>
</footer>
</div>
<?php wp_footer(); ?>
  </body>
</html>
```

Nachdem Sie alles Entscheidende über *style.css*, *functions.php*, *header.php* und *footer.php* wissen, kommen wir jetzt zur allerwichtigsten Datei.

2.2.4 index.php

Nur aufgrund der *index.php*-Datei funktioniert eine WordPress-Site überhaupt. Wie Sie in Abb. 2-3 bereits gesehen haben, ist das Fallback-Template für jede Template-

Datei die *index.php*. Das bedeutet, dass alle Beiträge, Seiten, CPTs, Archivseiten, Suchergebnisse, Autorenseiten oder andere Elemente standardmäßig den Code von *index.php* benutzen, wenn kein anderes Template existiert. Im Wesentlichen können Sie jeden Inhalt, den Sie im Administrationsbereich von WordPress eingegeben haben, anzeigen lassen, solange in Ihrer *index.php* der richtige Code steht.

In der *index.php*-Datei sollte es drei wichtige Bereiche geben: einen Aufruf der header-Funktion `wp_head()`, einen Aufruf der footer-Funktion, `wp_footer()` und den überaus wichtigen WordPress-Loop, den wir uns im nächsten Abschnitt ansehen.

Genauso wie bei jeder anderen Website hängt der Aufbau der *index.php*-Datei davon ab, was Sie auf der Site haben wollen; der große Unterschied zu anderen Websites ist allerdings, dass diese Datei wirklich den Aufbau aller Seiten der Website beeinflussen kann. Natürlich ist das im Normalfall nicht der Fall, weil Theme-Designer auch andere Templates erstellen, aber es ist etwas, was Sie auch nicht aus den Augen verlieren sollten.

Hinweis

Es ist gängige Praxis, über die *index.php* die Anzeige der neuesten Blogbeiträge zu steuern und eventuell auch die Anzeige der Suchergebnisse und der Archivseiten. Um die anderen Sachen kümmern sich eigene Template-Dateien.

Das Wichtigste, was Sie sich merken müssen, ist Folgendes: Wenn Sie eine angepasste Startseite haben möchten, dann sollten Sie dafür *nicht* die *index.php*-Seite nehmen. Würden Sie das nämlich machen, ist es ziemlich sicher, dass Seiten wie die Ergebnisse der Suche (*search.php*) oder Archivseiten (*archives.php*) nicht mehr funktionieren. Stattdessen sollten Sie ein Seiten-Template – das Sie vielleicht *page-home.php* nennen – erstellen oder meinen empfohlenen Ansatz nehmen und die WordPress-Template-Hierarchie verwenden. Sie können zwei Wege gehen, über *home.php* oder *front-page.php*. Was die Hierarchie anbelangt, so setzt sich *home.php* gegenüber *index.php* durch, aber nur bei der Startseite der Website. Wenn Sie also wollen, dass der Loop auf der Startseite anders aussieht als auf den anderen Seiten, dann erstellen Sie eine Datei namens *home.php* und WordPress kümmert sich um den Rest. Wenn Sie hingegen eine statische Startseite haben wollen (also eine, die nicht eine Liste der letzten Beiträge beinhaltet), so sollten Sie *front-page.php* nutzen. Die Art, wie WordPress die Datei *front-page.php* behandelt, können Sie sich durch folgende Schritte verdeutlichen:

1. Erstellen Sie zuerst ein *front-page.php*-Template in Ihrem Theme.
2. Legen Sie eine neue Seite im Administrationsbereich von WordPress an, beispielsweise mit dem Namen »Startseite«.
3. Gehen Sie dann zu *Einstellungen/Lesen* im Dashboard und ändern Sie *Startseite zeigt* von *Deine letzten Beiträge* in *Eine statische Seite*.
4. Wählen Sie dann die Startseite aus der Auswahlliste.

Jetzt wird unabhängig davon, welches Template die Startseite nutzt, *front-page.php* den Vorrang haben (Abb. 2-4).

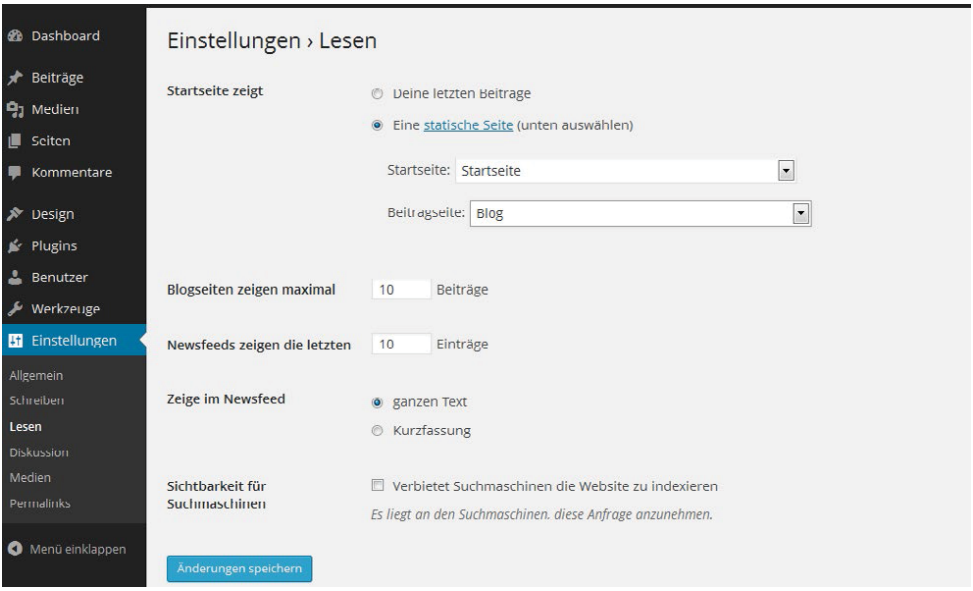

Abb. 2-4 *Die Einstellung Lesen im Administrationsbereich von WordPress, wo Sie eine statische Startseite auswählen können.*

Damit kennen Sie die Dateien, die meiner Meinung nach die wichtigsten in einem WordPress-Theme sind. Aber es fehlt noch eine Möglichkeit, den Inhalt, den Sie im Administrationsbereich eingegeben haben, auf Ihrer Website ausgeben zu lassen. Dafür haben wir den WordPress-Loop.

2.3 Der Loop

Der WordPress-Loop (der Loop oder auch *The Loop* genannt) ist nicht weniger als das Herz jeder WordPress-basierten Site. Er steuert, wie Inhalt in einem Template angezeigt wird, und deswegen müssen Sie einfach wissen, wie der Loop funktioniert. Hier sehen Sie, was im WordPress-Codex über den Loop steht:

> ***Der Loop*** *wird von WordPress benutzt, um jeden Beitrag anzuzeigen. WordPress verwendet den Loop, um jeden Beitrag zu bearbeiten, der auf der aktuellen Seite angezeigt werden soll, und formatiert ihn entsprechend bestimmter Kriterien im Loop-Tag. HTML- oder PHP-Code, der im Loop steht, wird bei jedem Beitrag wiederholt.*

Im Wesentlichen hat WordPress eine Reihe von Funktionen für folgende Zwecke:

1. Sicherstellen, dass es Beiträge gibt, die angezeigt werden können.

2. Diese Beiträge ausgeben lassen.
3. Besondere Funktionen, die Template-Tags genannt werden, erlauben es Ihnen genau anzupassen, wie die Inhalte aus den Beiträgen angezeigt werden. Der Inhalt wird aufgrund einer Anfrage ausgelesen, die an die Seite gesendet wird, die Sie aufrufen. Diese Abfrage wird aus der URL ausgelesen.

Seiten-URLs in WordPress

Standardmäßig sieht eine WordPress-URL etwa folgendermaßen aus: http://www.example.com?p=123. In diesem Beispiel ist p eine Variable, die die ID eines Beitrags darstellt.

WordPress hat eine unglaublich ausgefeilte Linkstruktur, die als Permalinks bekannt ist. Diese erlaubt es Ihnen, lesbare URLs zu erzeugen, die mehr Informationen beinhalten wie beispielsweise das Erscheinungsdatum des Beitrags, den Beitragstitel usw. Diese können Sie über den Administrationsbereich von WordPress über *Einstellungen/Permalinks* steuern. Der Codex bietet eine ausführliche Erläuterung dazu unter http://rwdwp.com/14.

Obwohl jeder Loop in einem bestimmten Template anders aussehen kann (beispielsweise können *single.php* und *page.php* unterschiedliche Loops haben), haben sie doch immer dieselbe Grundstruktur.

2.3.1 Der Grundaufbau

Erst einmal sollten Sie wissen, dass ein Template mehr als einen Loop haben kann. Es gibt einen Loop, der die erste Anfrage behandelt, die an die Seite weitergeleitet wird. Ich werde diesen Loop als »Haupt-Loop« bezeichnen und die anderen als »zweitrangige Loops«.

Der Haupt-Loop in einem Template beginnt folgendermaßen:

```
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
```

Was passiert hier? Drei WordPress-Funktionen werden aufgerufen:

- `have_posts()` stellt sicher, dass es Beiträge gibt, die angezeigt werden können. Wenn es keine Beiträge gibt, gibt `have_posts()` `false` zurück.
- Dieselbe Funktion soll kontinuierlich verfolgen, ob es Beiträge gibt, deswegen wird sie als Bedingung in einer `while`-Schleife verwendet.
- `the_post()` greift auf einen Beitrag aus einer Reihe von Beiträgen zu.

Noch einmal: Weil es der Haupt-Loop ist, mit dem wir es gerade zu tun haben, werden die Beiträge aufgrund der Anfrageparameter ausgewählt, die an die Seite übergeben werden. Sie könnten übrigens die ursprüngliche Abfrage mit `WP_Query()` oder `get_posts()` überschreiben, um angepasste Informationen zu erhalten, aber darüber reden

wir später. Unabhängig davon, solange die Abfrage Beiträge zurückliefert, gibt `have_posts()` `true` zurück und wir gelangen in den Loop.

Den Loop beenden Sie auf folgende Art:

```
<?php endwhile; else: ?> <?php _e('No posts were found. Sorry!'); ?> <?php endif; ?>
```

Hier haben wir einen einfachen Fallbackmechanismus in Form einer `if-else`-Anweisung. Wenn `have_pots()` `false` zurückgibt, erhält der Benutzer den Hinweis, dass es keine Beiträge gibt. Innerhalb des Loops können Sie die unterschiedlichsten Dinge tun, um die Anzeige der Beiträge genau anzupassen. Alle Template-Tags finden Sie im Codex und sie sind recht intuitiv benannt, sodass Sie einen Eindruck erhalten, was sie machen. Da sind Tags für den Titel (`the_title()`), den Zeitpunkt (`the_time()`), den Inhalt (`the_content()`), den Ausschnitt (`the_excerpt()`), die Kategorie (`the_category()`), die Tags (`the_tags()`), den Permalink (`the_permalink()`) und viele mehr. Es lohnt sich wirklich alle Template-Tags anzusehen, die es gibt, da Sie mit ihnen recht viel erreichen können.

Alle Template-Tags geben automatisch die Information aus, die sie zurückliefern. Wenn Sie hingegen wollen, dass diese Information nicht automatisch ausgegeben wird, können Sie andere Funktionen nutzen: Die meisten dieser Funktionen haben Gegenstücke mit `get`. Beispielsweise gibt `get_the_title()` den Wert nur zurück, anstatt ihn gleich auszugeben. Hier sind ein paar Template-Tags und ihre Funktion:

- `the_content()`: Das gibt den Inhalt des Beitrags aus, also das, was in den Beitragseditor auf der Administrationsseite eingegeben wurde. Bei einem Einzelseiten-Template (also beispielsweise *single.php*) wird der gesamte Inhalt ausgegeben. Auf einer Seite mit mehreren Beiträgen wird nur der Inhalt bis zu `<!--more-->` ausgegeben. Wenn `<!--more-->` nicht im Beitragsinhalt ergänzt wurde, wird der ganze Inhalt angezeigt.
- `the_excerpt()`: Diese Funktion gibt die ersten 55 Wörter eines Beitrags aus und ergänzt […] am Ende des Strings. Sie akzeptiert keine Argumente. Sowohl die Anzahl der Wörter als auch das Auslassungszeichen (…) können in der *functions.php*-Datei über Filter geändert werden.
- `the_permalink()`: gibt die absolute URL des Beitrags oder der Seite in dem Format aus, wie es im Administrationsbereich von WordPress unter *Einstellungen/Permalinks* festgelegt wurde.

2.3.2 Mehrfache Loops

Es gibt Fälle, in denen Sie neben dem Haupt-Loop einen zweiten Loop in Ihre Seite integrieren wollen – vielleicht um andere Beiträge anzeigen zu lassen, zusätzliche Inhalte, Bilder oder sonst etwas. Glücklicherweise bietet WordPress auch für diesen Fall eine Lösung.

Es gibt mehrere Wege, das zu realisieren; darunter auch einen, der ganz eindeutig als »der falsche Weg« gilt. Vielleicht ist Ihnen im Codex schon die Funktion `query_`

posts() begegnet; diese modifiziert in den meisten Fällen die Beiträge oder zeigt den Inhalt an, den Sie wollen. Allerdings verwendet auch der Haupt-Loop diese Funktion, deswegen kann es, wenn Sie sie aufrufen, zu Konflikten bei der Inhaltsausgabe kommen. Es kann passieren, dass Sie am Schluss mit zwei verschiedenen Titeln, dem falschen Inhalt oder anderen unvorhersehbaren Problemen dastehen.

Hinweis

Sogar auf der Codex-Seite zu query_posts() steht, dass Sie diese Funktion nicht verwenden sollten.

Glücklicherweise gibt es zwei bessere Arten, weitere Inhalte zu laden: WP_Query und get_posts(). Beide machen im Endeffekt dasselbe, aber ich werde get_posts() besprechen, weil es etwas leichter zu verstehen ist.

get_posts() erlaubt es, mehrere Loops in einem Template zu erstellen, ohne dass Sie die Hauptabfrage der Seite ändern müssen. Der einzige Unterschied ist, dass Sie Ihren Loop ein bisschen anders aufbauen müssen.

```
<?php
$custom_posts= get_posts(array('numberposts' => 4, 'category'=> 3, orderby =>
     'title'));
foreach ($custom_posts as $custom_post) :
setup_postdata($custom_post); ?>
<h3><?php the_title(); ?></h3>
<?php the_excerpt(); ?>
<?php endforeach; ?>
```

Vielleicht haben Sie den kleinen Unterschied bemerkt, wie wir diesen Loop geschrieben haben im Vergleich zum Haupt-Loop. Die folgende Zeile aus dem Haupt-Loop

```
<?php if (have_posts()) : while (have_posts()) :     the_post(); ?>
```

ist für das reserviert, was über query_posts() ermittelt wird, was die Standardinformation für die Seite sein sollte. Stattdessen liefert get_posts() ein Array von Beiträgen, die Sie in einer eigenen Variable speichern können, um sie dann bei jedem Loop zu durchlaufen.

Um die normalen Template-Tags zu nutzen, die wir uns angesehen haben, rufen Sie die Funktion setup_postdata() auf, übergeben ihr die Informationen aus dem aktuellen Beitrag – der jetzt in der Variable $custom_post gespeichert ist. Dann können Sie weiter wie gewohnt vorgehen und die üblichen Template-Tags nutzen.

2.4 Custom-Post-Types

Als die Custom-Post-Types (CPTs »Individuelle Inhaltstypen«) in WordPress 3.0 eingeführt wurden, dachte ich, sie wären die allerbeste Erfindung bei WordPress der letzten Zeit. Sie machten aus WordPress ein echtes Content-Management-System

(CMS), denn sie ermöglichten Webentwicklern, nicht nur Beiträge und Seiten zu ergänzen, sondern beliebige Inhalte.

Ein CPT gibt uns die Möglichkeit an die Hand, innerhalb von WordPress mit verschiedenen Inhaltstypen umzugehen. Jeder CPT wird genauso wie ein Beitrag oder eine Seite behandelt (Abb. 2-5).

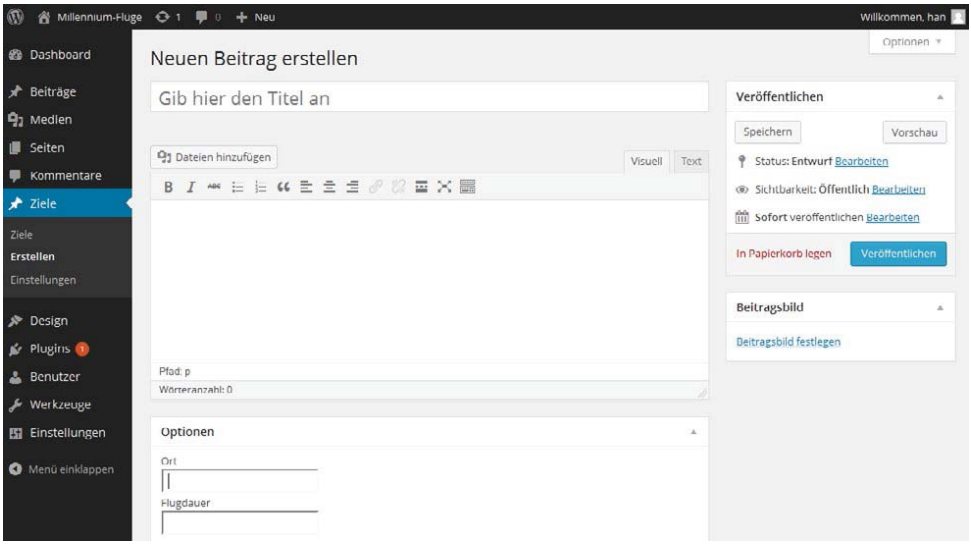

Abb. 2-5 *Der Administrationsbereich für einen CPT zum Thema »Ziele«, den ich für die Site Millennium Flüge entwickelt habe. Dieser CPT listet Informationen zu den angebotenen Reisezielen auf.*

Hinweis

Ein besserer Name als Custom-Post-Type wäre wahrscheinlich Custom Content Type, also angepasste Inhaltstypen gewesen, da Leute bei dem Wort »Post« an die Blog-Beiträge (Posts) denken.

CPTs haben dieselbe Basis wie die in WordPress integrierten Inhaltstypen, von denen es fünf gibt. Die meisten WordPress-Benutzer kennen Beiträge und Seiten und können diese als Inhaltstypen verstehen, weil sie so behandelt werden. Allerdings gibt es drei weitere Inhaltstypen:

- **Attachments** sind Dateien, die in WordPress über *Medien hinzufügen* hochgeladen werden, es können Bilder, Videos, PDFs und andere unterstützte Dateiformate sein.
- **Revisionen** sind Entwürfe oder verschiedene Versionen von Beiträgen oder Seiten.
- **Navigationsmenüs** sind Menüs, die über den Punkt *Design/Menüs* im WordPress-Administrationsbereich verwaltet werden.

CPTs erlauben unbeschränkt viele Inhaltstypen, sie werden für Branchenbücher, Personen, Immobilien, Kurse, Hausaufgaben und vieles mehr verwendet. In unserem Beispiel werden wir CPTs für die Ziele verwenden, zu denen man mit Millennium Flüge reisen kann.

Hinweis

Ein CPT für die Verwaltung von Immobilien könnte Titel, Beschreibung und ein Foto beinhalten (alles über den Standard-WordPress-Editor verfügbar), aber auch Ergänzungen für den Preis, die Adresse, die Anzahl von Zimmern und Bädern, Informationen über den Verkäufer, Größe in Quadratmetern usw.

Ich werde in diesem Kapitel nicht alle Details über die Programmierung von CPTs besprechen (dazu gibt es mehr in Kapitel 7), aber ich werde die grundlegende Funktionsweise erläutern und nützliche Tipps zu ihrer Erstellung geben.

Die Funktion, die für Sie CPTs erstellt, heißt `register_post_type()` und sie akzeptiert zwei Argumente: einen String für den Inhaltstyp, der als Slug verwendet wird, und ein Array von Argumenten für den Inhaltstyp. Die Argumente sind umfangreich, denn damit können Sie viel steuern – angefangen damit, eine einfache Beschriftung zu vergeben (wie beispielsweise »Ziele«) bis zur Änderung des Menü-Icons und der Position. Alle Details dazu finden Sie im WordPress-Codex unter http://rwdwp.com/15.

Wenn Sie sich entscheiden, dass Sie Kategorien oder Tags brauchen (die beide unter dem Begriff *Taxonomie* subsumiert werden), so müssen Sie dies zusätzlich zur Definition der CPTs angeben. Wenn Sie erst einmal einen CPT registriert haben, können Sie neue Taxonomien erstellen und WordPress anweisen, diese zu beliebigen Inhaltstypen hinzuzufügen – inklusive Ihrer selbst erstellten. Das machen Sie über die Funktion `register_taxonomy()`, die drei Argumente erwartet: einen Namen, ein Array von Inhaltstypen (die per Slug angegeben werden) und ein Array von Argumenten.

Bei der Entwicklung eines CPT würde ich Ihnen empfehlen, unbedingt Folgendes zu machen:

- Platzieren Sie Ihre CPTs in einem eigenen Plugin. Wie bereits erwähnt, sind CPTs schlicht und einfach Inhalt. Wenn Sie sie einem Theme hinzufügen (beispielsweise über die *functions.php*-Datei), dann verlieren Sie diese CPTs in dem Moment, in dem Sie das Theme deaktivieren.
- Planen Sie Ihre CPTs gut. Ich finde es hilfreich, mir die Felder aufzuzeichnen, d.h. wofür sie stehen werden (also beispielsweise Name -> Titel) und – das ist das Wichtigste – wie jedes Feld verwendet werden soll. Es ist beispielsweise wichtig, Textfelder von Kategorien und Tags zu unterscheiden. Wenn Sie eine falsche Implementierung wählen, müssen Sie vielleicht eine Lösung zusammenhacken oder Ihre CPTs neu programmieren.

- Wenn Sie können, sollten Sie ein einfaches Framework oder Template für Ihre CPTs erstellen. In den Quellen im Anhang finden Sie einen Link zu meinem Framework/Template bei GitHub. Ich habe es bei fast allen meinen CPTs verwendet und es hilft, den Entwicklungsprozess zu beschleunigen.

Nachdem Sie jetzt ein bisschen was über CPTs und darüber erfahren haben, dass sie in eigenen Plugins untergebracht werden sollen, sollten wir uns der Plugin-Erstellung zuwenden.

2.5 Plugins und Shortcodes

Plugins und Shortcodes bieten den Benutzern eine gute Möglichkeit, die Funktionalität zu erweitern, ohne dass sie wissen müssen, wie sie den eigentlichen Kern von WordPress programmieren oder verändern (was sie ja nicht tun sollten). Und es ist unsere Aufgabe als Webentwickler, dafür zu sorgen, dass unsere Plugins beim Anwender einen guten Eindruck hinterlassen. Legen wir los: Hier sind die Grundlagen der Plugin-Entwicklung.

Ein Plugin ist ein Programm oder eine Reihe von Funktionen, die die Funktionalität einer WordPress-Website erweitern. Ein Plugin kann alles Mögliche sein – von einem Einzeiler bis zu einem komplexen Programm mit mehreren Dateien, das aus WordPress eine E-Commerce-Anwendung macht. Alle Plugins werden im Ordner */wp-content/plugins* gespeichert. Und wenn sie korrekt definiert sind, erscheinen sie im WordPress-Administrationsbereich unter dem Punkt *Plugins* (Abb. 2-6).

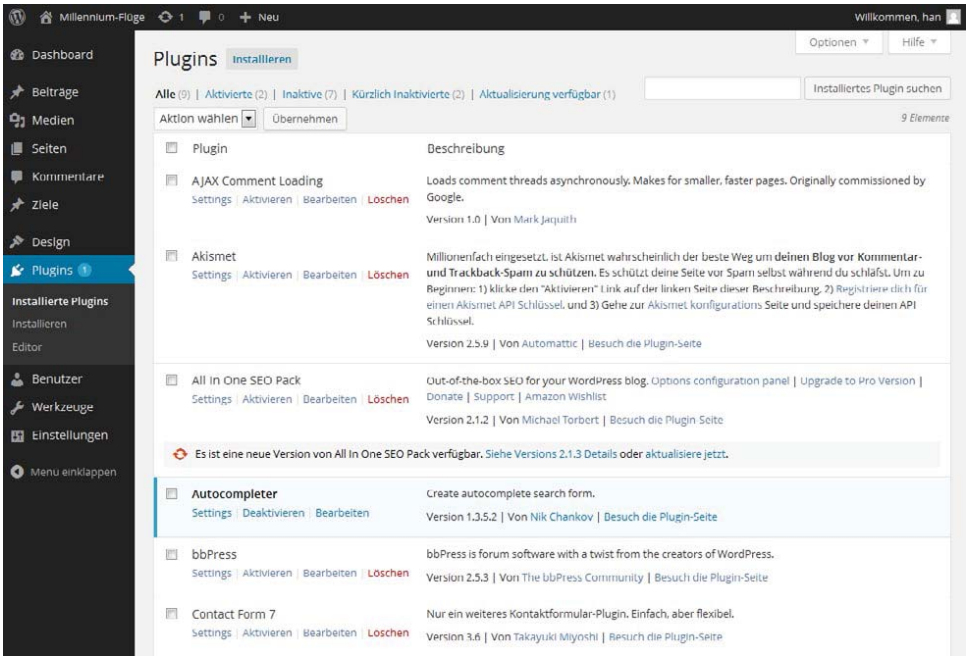

Abb. 2-6 *Der Plugin-Bereich von WordPress, wo Sie Plugins aktivieren, deaktivieren, hinzufügen oder löschen können.*

Jedes Plugin wird in einem eigenen Unterordner innerhalb von *wp-content/plugins/* gespeichert, den Sie beliebig benennen können. Es empfiehlt sich, dem Unterordner einen eindeutigen und gleichzeitig aussagekräftigen Namen zu geben, sodass es keine Konflikte mit anderen Plugins gibt und der Nutzer das Plugin sofort erkennt. Das Gleiche gilt auch für die Hauptdatei, die ich normalerweise genauso wie den Unterordner benenne. Für den Ziele-CPT habe ich das Unterverzeichnis */mf-destinations-cpt/* genannt (und parallel dazu heißt die Hauptdatei *mf-destinations-cpt.php*). Wie auch bei Funktionen und Klassen habe ich meine Plugin-Informationen mit einem Präfix versehen, was die Wahrscheinlichkeit für Konflikte erheblich reduziert. Wir werden das Präfix *mf-* für all unsere Dateien, Ordner, Funktionen, Klassen und Shortcodes nutzen.

In der Hauptdatei Ihres Plugin-Unterordners steht die Definition des Plugins, ganz ähnlich wie Sie auch die Theme-Definition in Ihrer Haupttheme-Datei im Theme-Unterordner angeben:

```
/*
Plugin Name: Ziele-CPT
Plugin URI: http://millenniumflights.com
Description: Dieses Plugin erstellt einen Custom-Post-Type und eine Template-Seite
für alle Ziele, die Millennium Flüge anbietet.
Author: Joe Casabona
Version: 1.0
Author URI: http://casabona.org/
*/
```

Sie können auch noch die verwendete Lizenz angeben (meist wird die GPL eingesetzt) und auch den Wortlaut der Lizenz, wenn Sie möchten.

Über die GPL

In den letzten Jahren gab es eine heiße Diskussion über WordPress-Themes, Plugins, Entwickler und die GPL, was für GNU General Public License steht. Die GPL ist der Grund dafür, dass WordPress Open-Source-Software ist. Es gibt eine Klausel in der GPL, in der es heißt, dass alle Dateien, die WordPress-Code verwenden, ebenfalls unter der GPL veröffentlicht werden müssen. Das bedeutet, dass jedes Plugin oder Theme, das Sie herausgeben, zu größten Teilen Open-Source sein müsste. Aber es gibt eine kleine Hintertür.

Weil CSS-Dateien und Bilder keinen WordPress-Code nutzen, können diese unter einer beliebigen Lizenz veröffentlicht werden. Das hat zu Konflikten zwischen Puristen, die der Meinung sind, dass die vollständigen Themes und Plugins Open-Source sein sollten, und anderen Entwicklern geführt, die eine duale Lizenz verwenden: GPL für WordPress-Code und eine proprietäre für ihren CSS-Code und ihre Bilder.

Der Initiator von WordPress, Matt Mullenweg, vertritt deutlich die Meinung, dass alle im Zusammenhang mit WordPress stehenden Projekte vollständig Open-Source sein sollten.

Wenn das Plugin definiert ist, können Sie mit der Programmierung loslegen! Es gibt ein paar Konventionen, die Entwickler beachten sollten, dazu gehört alles, was auf den WordPress-Coding-Standard-Seiten aufgeführt ist (http://rwdwp.com/16). Das beinhaltet viel, worüber wir bereits geredet haben – wie Präfixe zu benutzen, um Konflikte zu vermeiden –, aber auch allgemeine Empfehlungen für einen guten Programmierstil (wie die Überprüfung und Bereinigung von Daten, bevor sie in der Datenbank gespeichert werden).

Die Plugin-API erlaubt es uns, die mächtigen Funktionen von WordPress anzuzapfen, indem wir Hooks, Actions und Filter nutzen.

2.5.1 Hooks, Actions und Filter

Hooks sind Code-Schnipsel, die uns ermöglichen, gut mit WordPress zusammenzuarbeiten; es gibt zwei Arten von Hooks: Action-Hooks und Filter-Hooks.

Action-Hooks sind Funktionen, die mit irgendeinem Ereignis verbunden sind, das in WordPress ausgelöst wird. Beispielsweise können Sie eine Aktion hinzufügen, die automatisch einen Tweet versendet, wenn Sie einen Beitrag schreiben, indem Sie den Hook `publish_post` verwenden. Nehmen wir an, Sie haben die fiktive Funktion `mf_send_tweet()`. In Ihrem Plugin (normalerweise direkt vor oder nach der Funktionsdefinition) können Sie dann den folgenden Code hinzufügen:

```
add_action('publish_post', 'mf_send_tweet');
```

Das weist WordPress an, in dem Moment, in dem der Hook `publish_post` ausgelöst wird, die Funktion `mf_send_tweet()` aufzurufen. Es gibt eine unglaublich lange Liste von Action-Hooks, die Sie alle im Codex unter http://rwdwp.com/17 finden.

Ähnlich erlauben Ihnen Filter, aufgrund eines ausgelösten Hooks den Inhalt zu verändern. Im Wesentlichen reicht ein Filter einen Inhalt an eine von Ihnen bestimmte Funktion weiter. Dann ändern Sie den Inhalt und geben ihn wieder zurück, damit er auf der Website ausgegeben werden kann. Sehen wir uns beispielsweise den folgenden Code an:

```
function mf_add_signature($content){
$content .= '<p><img src=»'. IMAGES .'/sig.png" alt="Joseph L. Casabona, Gründer"
     /></p>';
return $content;
}
add_filter( "the_content", "mf_add_signature" );
```

Hier haben wir eine Funktion, die eine Signatur mit einem Bild am Ende des Inhalts ergänzt, der an die Funktion übergeben wird. Wichtig dabei: Sie müssen den Inhalt mit `return` zurückgeben. Anderenfalls funktioniert der Filter nicht richtig.

Wenn Sie die Funktion definiert haben, können Sie einen Filter wie im Beispiel in der letzten Zeile über die Funktion `add_filter()` hinzufügen. Genauso wie bei `add_action()` können Sie zwei Argumente angeben: den gewünschten Hook und die Funktion,

die aufgerufen werden soll, wenn der Hook ausgelöst wird. Eine Liste aller Filter, die Sie benutzen können, finden Sie unter http://rwdwp.com/18.

Lassen wir Hooks, Actions und Filter einmal beiseite: Vielleicht wollen Sie Plugin-Daten für die spätere Verwendung speichern. Für das Speichern von Daten haben Sie vier Möglichkeiten:

1. Sie können das Speichern über CPTs durchführen, wie Sie es auch in Ihrem Plugin machen.
2. Sie können WordPress-Optionen-Funktionen verwenden, die Ihnen im Wesentlichen erlauben, Variablen zu definieren und in der *wp_options*-Datenbanktabelle zu speichern. Das ist die empfohlene Art für Theme-Optionen, so lange Sie eine relativ kleine Menge an Daten speichern.
3. Sie können selbstdefinierte Taxonomien einsetzen, die Sie ebenfalls in Ihrem Plugin verwenden.
4. Außerdem können Sie eine eigene Datenbanktabelle erstellen. In den meisten Fällen sollten die drei erwähnten Wege gut funktionieren, aber Sie haben auch die Möglichkeit, eine Datenbanktabelle zu erzeugen. Wenn Sie das machen, sollten Sie unbedingt die Anweisungen aus dem Codex unter http://rwdwp.com/19 befolgen.

Schließlich sollten Sie unbedingt Shortcodes und Template-Tags für die Hauptfunktionalität Ihrer Plugins definieren und dokumentieren. Sie wollen schließlich nicht, dass die Benutzer sich durch Ihren Code wühlen müssen, um die gesuchten Funktionalitäten zu finden. Template-Tags sind unkompliziert; sie sind einfache Funktionen, die der Benutzer aufrufen kann (was in PHP jede Funktion ist, die nicht explizit als `private` oder `protected` deklariert ist). Benutzer können diese Funktionen in ihren eigenen Plugins oder Themes verwenden. Sie als Entwickler können diese Template-Tags dann einsetzen, um Shortcodes zu erzeugen.

2.5.2 Shortcodes

Shortcodes sind Schnipsel, die der Benutzer in seinen WordPress-Editor einfügen kann, um eine bestimmte Funktionalität aufzurufen. Shortcodes haben immer dasselbe Format `[name-des-shortcodes]`. WordPress machte Ihnen ihre Erstellung recht einfach. Gehen wir einmal von der folgenden Funktion aus:

```
function mf_hello_world(){
  return "Hallo Welt!";
}
```

Wenn Sie jetzt einen Shortcode erstellen wollen, der »Hallo Welt« in einen Beitrag oder eine Seite einfügt, dann müssen Sie nur irgendwo in Ihrem Plugin (wahrscheinlich entweder oberhalb oder unterhalb der Funktionsdefinition) Folgendes schreiben:

```
add_shortcode('mf_hello', 'mf_hello_world');
```

Damit ist der Shortcode [mf_hello] erstellt, der Ihre Funktion aufruft, wenn er im Inhalt verwendet wird. Sie werden feststellen, dass es besser ist, den Text mit return zurückzugeben als mit print auszugeben. Der Grund hierfür ist, dass die Informationen, die von der Funktion zurückgegeben werden, im Inhalt eingefügt werden. Eine Ausgabe im Shortcode selbst wird nicht funktionieren, weil sie wahrscheinlich an der falschen Stelle erfolgen würde.

Sie können mit Shortcodes und dem Einsatz von Argumenten und anderen Dingen weit kommen. Wir werden uns ein paar Beispiele dafür in Kapitel 7 ansehen.

2.6 Zusammenfassung

Puh, was für ein Kapitel! Wir haben uns WordPress-Themes angesehen, CPTs, Plugins und Shortcodes und ein paar zusätzliche Informationen, die Sie zum Loslegen brauchen. Dabei haben wir aber wirklich nur an der Oberfläche gekratzt. Wenn Sie tiefer in die Entwicklung von WordPress-Themes einsteigen wollen, finden Sie am Ende des Buches ein paar nützliche Quellen aufgelistet, darunter auch mein voriges Buch. Was Sie sich merken sollten: Es ist wichtig zu unterscheiden zwischen dem, was in ein Theme, und dem, was in ein Plugin gehört.

Ich weiß, dass wir in diesem Kapitel nicht so viel gecoded haben, aber das wird sich ändern. In den nächsten Kapiteln werden wir ein Theme für Millennium Flüge entwickeln und dabei Optimierungen vornehmen, die WordPress uns für die Umsetzung besserer responsiver Designs bietet.

2.6.1 Fragen

1. Was sollten Sie sich selbst für eine Frage stellen, um zu entscheiden, ob ein Code besser im Theme oder in einem Plugin aufgehoben ist?
2. Welche Funktion sollten Sie nie nutzen, wenn Sie die Inhalte für einen Loop ermitteln wollen?
3. Was sind die fünf in WordPress integrierten Inhaltstypen?
4. Was ist der Unterschied zwischen Actions und Filtern?

2.6.2 Antworten

1. Möchte ich diese Funktionalität bei einem Redesign der Website behalten?
2. query_posts()
3. Beiträge, Seiten, Revisionen, Attachments und Navigationsmenüs
4. Actions lösen eine bestimmte Funktion aus, die ausgeführt werden soll. Filter hingegen ändern Inhalte in Abhängigkeit vom aufgerufenen Hook.