

7.1 Hintergrundwissen zu Themes

Von Themes war in diesem Buch schon oft die Rede, allerdings wurden sie weder verändert noch neu geschrieben; das soll sich jetzt ändern.

Wie Sie wissen, müssen sich die WordPress-Themes in einem Ordner im Verzeichnis `wp-content/themes/` befinden. In diesem Ordner liegen alle benötigten Dateien, das Stylesheet, die Templates, Bilder und eventuell weiterhin notwendige Daten, zum Beispiel Skripte. Der Ordner sollte genauso heißen wie das Theme. Zwei Dateien müssen in jedem Theme mindestens vorhanden sein: das Template `index.php` als Ausgangsdatei und das Stylesheet `style.css`.

In den Templates werden mit Hilfe der Template Tags die Daten aus der Datenbank generiert und auf der Webseite dargestellt. Das Haupt-Stylesheet `style.css` gibt Ihnen die Kontrolle über das Design der Webseite, zum Beispiel wie groß die Schrift sein soll und wie einzelne Elemente positioniert werden. Prinzipiell könnte man Styleangaben auch in HTML, also in den Templates machen, allerdings führt dies zu sehr unübersichtlichem Code, und die Änderungen sind nur schwer zu kontrollieren. Auch könnte man nicht mit einem Style alle Templates formatieren. Nach den Regeln des modernen Webdesign werden die Styleangaben deshalb in eine eigene Datei geschrieben.

Damit das Theme übersichtlich bleibt, besteht es typischerweise aus mehreren PHP-Dateien. Die folgenden vier Dateien nehmen einen besonderen Status ein, denn sie werden von den *Include Tags* unterstützt und müssen nicht über den kompletten Pfad eingebunden werden.

`header.php`

Das Header-Template, das sämtliche Informationen enthält, die sich im Bereich `head` des HTML-Dokuments befinden.

`sidebar.php`

Template für den Sidebar, für die Darstellung von Navigation und allen Inhalten, die übergreifend zur Verfügung stehen sollen, typischerweise im rechten Teil des Blog.

`footer.php`

Template für den Bereich Footer, die Fußzeile des Blog.

`comments.php`

Template der Kommentare, für die Einbindung der Kommentarfunktion. Ist es nicht im Theme vorhanden, wird es aus dem Standard-Theme geholt.

Die Include Tags sind wiederum Funktionen, die das Einbinden dieser Dateien erleichtern. Zur Verdeutlichung hilft ein kurzer Blick auf den Code, der sie definiert, zum Beispiel für den Header:

```
function get_header() {
    do_action( 'get_header' );
    if ( file_exists( TEMPLATEPATH . '/header.php' ) )
        load_template( TEMPLATEPATH . '/header.php' );
    else
        load_template( ABSPATH .
                        'wp-content/themes/default/header.php' );
}
```

Anhand dieser Funktion sehen Sie auch die Alternative. Sollte die Datei im Theme nicht vorhanden sein, so wird sie aus dem Default-Theme gezogen – auch ein Grund, warum das Theme immer in der Installation erhalten bleiben sollte.

Damit ergeben sich folgende vier Include Tags, um die vier Dateien einzubinden.

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

Alle weiteren Dateien eines Themes werden wie folgt eingebunden:

```
<?php include (TEMPLATEPATH . '/example.php'); ?>
```

Dabei muss die Datei lediglich im Theme-Verzeichnis liegen. Über die Konstante `TEMPLATEPATH` wird das entsprechende Verzeichnis je nach aktiviertem Theme gewählt.

Steht eine Datei außerhalb des Theme-Ordners (`TEMPLATEPATH`), was möglichst nicht vorkommen sollte, so kann sie mit einer absoluten Pfadangabe eingebunden werden:

```
<?php include (ABSPATH . 'wp-content/themes/default/header.php');
```

Nun haben Sie sechs Templates kennengelernt, die eine wichtige und übergeordnete Rolle spielen. WordPress unterstützt aber eine ganze Reihe weiterer Templates, die auch ohne das Hinterlegen der entsprechenden Kommentare im Template als WordPress-Templates erkannt werden. Alle möglichen Templates, die WordPress unterstützt und die bisher noch nicht genannt wurden, werden in der folgenden Liste ein näher erläutert.

`comments-popup.php`

Das Popup-Kommentar-Template. Ist auch dieses nicht präsent, wird es aus dem Standard-Theme geladen, insofern es im verwendeten Theme vereinbart ist.

`home.php`

Das Template für eine Startseite, wenn diese benötigt wird. Dieses Template ist nicht mit `index.php` zu verwechseln, es wird ausschließlich geladen, wenn die Startseite angesprochen wird, `index.php` hingegen immer dann, wenn ein Template nicht vorhanden ist.

`single.php`

Das Single-Template sorgt für die Darstellung eines einzelnen Beitrags. Ist dieses Template nicht vorhanden, wird `index.php` genutzt.

`page.php`

Das Seiten-Template für die Darstellung einer statischen Seite.

`category.php`

Das Kategorien-Template wird immer gezogen, wenn es sich um den Verweis auf eine Kategorie handelt.

`author.php`

Das Autoren-Template sorgt für die Darstellung der Informationen zu einem Autor.

`date.php`

Das Datum/Zeit-Template wird gezogen, wenn die Datenbank Inhalte zu Jahr, Monat, Tag, Stunde, Minute oder Sekunde holen soll.

`archive.php`

Das Archiv-Template wird immer gezogen, wenn die Templates für Kategorie, Autor, Datum oder Zeit nicht vorhanden sind.

`search.php`

Das Such-Template erscheint, wenn eine Suche durchgeführt wurde.

`404.php`

Das „404 nicht gefunden“-Template erscheint, wenn die Adresse nicht vorhanden ist, und gibt die entsprechende Information aus.

Das Vorhandensein jedes Templates kann mit Hilfe der Conditional Tags abgefragt werden. Dies ist wichtig, wenn Sie eigene Templates erstellen und die Ausgabe von den jeweiligen Funktionen abhängig machen wollen. Eine Übersicht und näherer Erläuterungen zu diesen Tags finden Sie weiter hinten in diesem Kapitel auf Seite 129.

Sie können weitere Templates einfügen, die nicht der WordPress-Vereinbarung entsprechen. Diese müssen über den Code eines anderen Templates eingebunden oder explizit in der Administrationsoberfläche vereinbart werden.

Templates, die im Bereich **Schreiben** auswählbar sein sollen, müssen einen entsprechenden Kommentarbereich im Code mit vereinbarten Schlüsseln

enthalten. Dazu mehr im übernächsten Abschnitt. Die Hierarchie können Sie detailliert im Codex von WordPress nachlesen.¹

7.1.1 Stylesheet

Damit WordPress das Stylesheet erkennt und verarbeiten kann, müssen neben den CSS-Style-Informationen eine Reihe von Informationen in einem Kommentarfeld hinterlegt werden, und die Datei muss `style.css` heißen. Im Folgenden ein Beispiel für die Vereinbarung im Stylesheet. Die Schlüssel innerhalb des Kommentarbereichs müssen stimmen, da WordPress die Werte andernfalls nicht auslesen kann. Im Anschluss an die Schlüssel können Sie aber weitere Informationen ergänzen. Achten Sie darauf, dass Sie dabei nicht die Schlüsselfelder (mit `:`) benutzen.

```
/*
Theme Name: Example
Theme URI: URL, wo es Informationen zum Theme gibt
Description: eine kurze Beschreibung
Author: Name des Autors
Author URI: URL des Autors
Template: Sollte das Theme auf einem anderen basieren, dann können Sie
das hier das vermerken - optional
Version: die Versionsnummer Ihres Themes

Zusätzliche Informationen, Danksagungen oder Lizenz-Infos.
*/
```

Im Anschluss an diesen Kommentarblock beginnen die Style-Definitionen, ganz nach den Layoutregeln der Cascading Stylesheets (CSS). Weitere optionale Stylesheets können jeden beliebigen Namen haben und werden in den Templates mit der entsprechenden HTML-Anweisung geladen (siehe Seite 124). Achten Sie aber darauf, dass nur die Datei `style.css` den Kommentarblock mit den WordPress-Vereinbarungen enthält.

7.1.2 Template-Dateien

Durch die Template-Dateien haben Sie die Möglichkeit, jede Seite individuell aufzubauen. Sie können jederzeit mit sehr wenig Aufwand ein weiteres Template hinzufügen, um damit bestimmten Anforderung gerecht zu werden. Ein kleines Beispiel soll dies verdeutlichen.

Die statischen Seiten eines WordPress-Blogs enthalten im Gegensatz zu Beiträgen typischerweise Inhalt, der keiner chronologischen Ordnung unterliegt und immer über die Navigation erreichbar sein soll. Jetzt soll, warum

¹ http://codex.wordpress.org/Template_Hierarchy

auch immer, ein Template es ermöglichen, solche Seiten ohne Überschrift zu generieren. Im Standard ist die Überschrift im Editor im Titel-Textfeld hinterlegt. Dieser Inhalt wird mit dem Template Tag `the_title()` ausgelesen. In den meisten Themes wird der Titel mit dem `h2`-Tag versehen, so dass es im Template folgendermaßen aussehen kann.

```
<h2><?php the_title(); ?></h2>
```

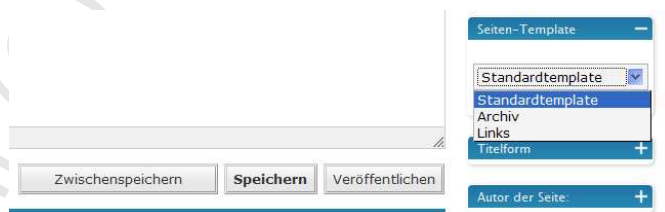
Dieser kleine String aus HTML und PHP soll im verwendeten Template nicht geladen werden. Dazu kopieren Sie die bisherige `page.php` und vergeben einen neuen Namen – z. B. `page_ohne_h2.php`

Diese neue Datei muss nun mit einigen Zeilen Kommentar ausgestattet werden, so dass WordPress das Template erkennt und innerhalb des Editorbereichs anbietet. Fügen Sie dazu folgende Zeilen oberhalb des HTML- und PHP-Codes ein.

```
<?php
/*
Template Name: page ohne h2
*/
?>
```

Sie können innerhalb dieses Bereichs natürlich noch mehr Informationen hinterlegen, jedoch sind diese für WordPress nicht von Bedeutung. Anhand des String `Template Name` erkennt WordPress dieses Template und bietet es im Bereich des Editors zur Auswahl an.

Abbildung 7.1:
Templateauswahl im
Bereich Seite
Schreiben



Damit das Template nun den Titel weglässt, müssen Sie nur noch den genannten Bereich löschen bzw. auskommentieren:

```
<!--<h2><?php the_title(); ?></h2-->
```

Erstellen Sie nun eine neue Seite im Editor des Administrationsbereichs und benutzen Sie Ihr neues Template, indem Sie es im Bereich **Seiten-Template**s auswählen. Nachdem Sie die Seite veröffentlicht haben, können Sie sich das Ergebnis im Blog ansehen. Die Überschrift (der Titel) fehlt in der Ausgabe.

Man kann also für wiederkehrende Vorlagen sehr schnell Templates erzeugen, statt alles per Hand im Editor zu realisieren.

7.1.3 Der Loop

Der Loop ist im Template die zentrale Ereignisschleife zur Darstellung von dynamischen Inhalten. Eine ganze Reihe von Template Tags (siehe Seite 126) funktionieren nur innerhalb des Loop. Möchten Sie Template Tags außerhalb des Loop anwenden, dann prüfen Sie gegebenenfalls, ob das funktioniert. Dazu steht Ihnen der Codex² von WordPress zur Verfügung. Durch den Loop kann die Applikation die Artikel so darstellen, wie es die Template Tags innerhalb der Templates vorgeben. Jedes Tag, ob HTML oder PHP, das sich innerhalb des Loop befindet, wird für jeden Beitrag angewandt. Wenn Sie ein Theme verändern oder erstellen und die Template Tags benutzen, dann müssen diese innerhalb des Loop stehen.

Typische Template Tags für die Verwendung innerhalb des Loop sind der Titel (`the_title`), die Zeit (`the_time`) und der Beitrag (`the_content`).

Der Loop beginnt mit folgendem Code:

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

Und endet mit diesen Zeilen:

```
<?php
        endwhile;
    else :
?>
<p><?php _e('Sorry, no posts matched your criteria.');" ?></p>
<?php
    endif;
?>
```

Eine Minimalvariante der `index.php` inklusive Loop sieht folgendermaßen aus:

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

² <http://codex.wordpress.org/>

```
        the_content();

    endwhile;
endif;
get_sidebar();
get_footer();
?>
```

Dem Aufruf der Datei `header.php`, die die HTML-Tags für Head und Body enthält, folgt der Loop inklusive dem Template Tag zum Aufruf des Beitrags `the_content`; den Abschluss bildet der Aufruf von Sidebar und Footer, der die schließenden Tags der HTML-Seite enthält.

Mit diesen wenigen Zeilen könnte WordPress laufen, der Loop ist vorhanden und das Template Tag holt in Verbindung mit `the_content` den Inhalt des aktuellen Beitrags aus der Datenbank. Innerhalb des Loop können Sie natürlich viele weitere Template Tags verwenden, ebenso auch HTML.

7.2 Theme erweitern

Zumindest zu Beginn ist es eine gute Idee, bestehende Themes nach den eigenen Vorstellungen zu erweitern, ohne das Original zu zerstören. Das hat außerdem den Vorteil, dass Sie nach einem Update des zugrunde liegenden Themes Ihre Modifikationen nicht neu in das Theme schreiben müssen. Die neue Version kann problemlos eingespielt werden und Ihre Anpassungen bleiben erhalten.

Mit Hilfe eines zusätzlichen Stylesheets, also einer weiteren CSS-Datei, bringen Sie neue Styleinformationen in das Theme. Diese Datei wird mit Hilfe von HTML im entsprechenden Template, in der Regel `header.php`, eingebunden, so dass sie zusammen mit dem Original-Stylesheet geladen wird. Zusätzlich bekommt der `body`-Tag eine Klasse zugeordnet, die mit Hilfe des neuen Stylesheets angesprochen wird. Über diese `class` werden die neuen Styleinformationen umgesetzt.

Im Folgenden der Ablauf, damit Sie Idee und Lösung schnell und einfach nachvollziehen können.

Stylesheet hinzufügen

Erstellen Sie eine Stylesheet-Datei und speichern Sie diese unter dem Namen `custom.css` im Ordner des Themes. Achten Sie darauf, dass Sie nicht die Schlüsselwörter der `style.css` verwenden. Deshalb im Beispiel auch die Erweiterung vor jeder Information im Kommentarabschnitt um das Wort „Custom“. Der Kommentarbereich wird nicht benötigt, aber es ist hilf-

reich, wenn Sie sich eine saubere Arbeitsweise angewöhnen, und dazu zählt die Hinterlegung von Autor, Version und URL für weitere Kontakte und Informationen.

```
/*
Custom Theme Name: Meine Style-Daten
Custom Theme URI: http://example.com/
Custom Description: Erweiterung des Theme WordPress Standard DE-Edition
Custom Version: 0.1
Custom Author: Ihr Name
Custom Author URI: http://example.com/
*/
```

Theme modifizieren

Damit die Informationen aus dem zusätzlichen Stylesheet auch geladen werden, muss der `head`-Bereich des Themes erweitert werden. Fügen Sie dazu in `header.php` Ihres Themes den folgenden Code im `head`-Bereich ein, also zwischen dem öffnenden `<head>`- und schließenden `</head>`-Tag.

```
<link rel="stylesheet"
      href="<?php bloginfo('template_url'); ?>/custom.css"
      type="text/css" media="screen" />
```

WordPress verwendet XHTML 1.0 zur Darstellung, deshalb ist auch das Einfügen des Link-Tag nach dessen Regeln vorzunehmen. Die Adresse zur Datei `custom.css` wird im Attribut `href` hinterlegt. Dabei wird mit Hilfe der Template Tags die URL des verwendeten Themes angesprochen und die entsprechende Datei, `custom.css`, geladen. Wenn Sie sich den Quellcode Ihres Blogs nach der Modifikation ansehen, so werden Sie erkennen, dass die komplette URL zum Theme-Ordner im Attribut `href` steht. Sollten Sie also einmal das Theme unter einer anderen URL betreiben oder lokal testen, dann funktioniert der Verweis auf das zusätzliche Stylesheet weiterhin.

Anschließend muss der Body Ihrer Webseite eine eigene Klasse bekommen, über die dann die Styles angesprochen werden.

Suchen Sie in `header.php` den `<body>`-Tag und erweitern Sie ihn folgendermaßen:

```
<body class="custom">
```

Speichern Sie die Datei `header.php` und kopieren Sie sie in Ihr Theme.

Diese beiden Veränderungen am Theme müssen Sie natürlich nach jedem Theme-Update wieder durchführen, da die `header.php` überschrieben werden könnte. Aber es ist nur eine Kleinigkeit im Verhältnis zu einer kompletten und fehleranfälligen Theme-Modifikation im Original-Stylesheet.

Implementieren Ihrer Styles

Nun können Sie nach Belieben `custom.css` erweitern und das Theme nach Ihren Wünschen anpassen. Dafür gehen Sie folgendermaßen vor: Suchen Sie in `style.css` die entsprechende Formatierungssyntax, die Sie verändern wollen, und kopieren Sie den Abschnitt in die Datei `custom.css`. Ein Beispiel anhand des „WordPress Standard DE-Edition“-Themes soll dies verdeutlichen.

Im Stylesheet (`style.css`) des Themes finden Sie die Anweisung für die Schriftgröße im Content-Bereich.

```
#content {  
    font-size: 1.2em  
}
```

Diesen Abschnitt kopieren Sie und fügen ihn in die Datei `custom.css` ein. Danach nehmen Sie noch folgende Veränderung am Code vor:

```
.custom #content {  
    font-size: 2.2em;  
    color: red;  
}
```

Mit dieser Information verändern Sie das Erscheinungsbild des Blogs erheblich. Die Anweisung greift ebenso auf die ID `content` zu, allerdings nur im Bereich der Klasse `custom`, die über den ganzen `body` gilt, also das komplette Erscheinungsbild umfasst. Im Weiteren wird die Schriftgröße um 1em erhöht (em ist eine relative Einheit, 1em entspricht, grob gesprochen, der Standardgröße des Browsers ohne jegliche Formatierungen) und die Farbe des Textes wird rot.

Übrigens: Möchten Sie derartige Veränderungen live sehen und damit einfach und anschaulich nachvollziehen, dann arbeiten Sie am besten mit dem Browser Mozilla Firefox und dem Add-on „Web Developer“ (siehe Seite 167).

7.2.1 Template Tags

Mit Hilfe von Template Tags werden die Information in das Frontend von WordPress, also auf das Webblog, geladen. Template Tags sind PHP-Funktionen, die als kleine Code-Schnipsel in die Templates integriert werden und die eine Funktion aus den Core-Dateien von WordPress aufrufen. Die Parameter der Template Tags sind Variablen, die die Funktion benötigt, um richtig ausgeführt zu werden. Sie bekommen schon vordefinierte Werte zugewiesen, damit müssen nur die Parameter übergeben werden, die man auch wirklich verändern will.