

Clemens Gull

Know-how
ist blau.

PHP für WordPress

Themes und Templates selbst entwickeln

- > WordPress verstehen, anpassen und individuell ergänzen
- > WordPress optisch und funktional erweitern
- > Ausführliche PHP- und WordPress-Befehlsreferenz zum Nachschlagen

FRANZIS

Inhaltsverzeichnis

1	Einführung.....	9
1.1	Installation von Apache, PHP und mySQL.....	9
1.1.1	Die notwendigen Programme	10
1.1.2	Die Installation für Windows XP.....	11
1.1.3	Die Installation für Windows Vista oder Windows 7.....	14
1.1.4	Testen des Webservers und PHP bei Windows-Systemen	19
1.1.5	Die Installation für Mac OS X.....	21
1.2	Installation von Wordpress	37
1.2.1	Installation der PHP-Anwendung.....	37
1.2.2	Zugriffsrechte für Mac OS X.....	38
1.2.3	Installieren der Datenbank.....	40
1.2.4	Einrichten von Wordpress	42
1.3	Installation der Entwicklungsumgebung.....	49
1.3.1	Download der Software.....	50
1.3.2	Installation für Windows-Systeme	51
1.3.3	Installieren der Plug-Ins für Eclipse	53
1.3.4	Anpassen der Perspektive von Eclipse	58
1.4	Der Browser	60
1.4.1	FireBug	61
1.4.2	WebDeveloper	62
1.4.3	GridFox	63
2	Themen in Wordpress	67
2.1	Der Hintergrund zu Wordpress	67
2.1.1	Aufbau	67
2.1.2	Terminologie von Wordpress.....	69
2.1.3	Die Hierarchie von Templates.....	73
2.1.4	Der Loop	74
2.2	Themen.....	76
2.2.1	Aufbau eines Themas.....	76
2.2.2	Include-Tags	77
2.3	Ein neues Thema.....	78
2.3.1	Ein Projekt in Eclipse anlegen	78

2.3.2	Die Struktur des Themas anlegen.....	83
2.3.3	Grundlagen der Programmierung.....	85
2.3.4	Die Startdatei des Themas	101
2.4	Verfeinern der Startdatei	141
2.4.1	Den HTML-Kopf anpassen	141
2.4.2	Eine Variablenliste in PHP	148
2.4.3	Die Seitenleiste ergänzen.....	151
2.4.4	Zusätzliche Funktionen im Blog.....	166
2.4.5	Ein Template für die Kommentare.....	168
2.5	Zusätzliche Bereiche des Blogs.....	201
2.5.1	Das Template für einen einzelnen Artikel	201
2.5.2	Den Blog durchsuchen	206
2.5.3	Das Template für die Seiten	215
2.5.4	Das Stylesheet verbessern	217
2.5.5	Die zweite Seitenleiste einfügen	221
2.5.6	Einfügen der Artikelnavigation	227
2.5.7	Einzelansicht des Artikels anpassen	231
2.5.8	Die Ansicht der Seite anpassen.....	238
2.6	Das Archiv des Blogs.....	240
2.6.1	Das Template für das Archiv.....	240
2.6.2	Das Archiv formatieren.....	253
2.7	Die Fehlerseite des Blogs erstellen.....	254
2.7.1	Die Programmierung von 404.php	254
2.7.2	Die Fehlerseite formatieren	261
3	Ein Thema erweitern	265
3.1	Ein Thema verändern	265
3.1.1	Übersetzen eines Themas	265
3.1.2	Eine Sprachdatei erstellen	291
3.1.3	Layout anpassen.....	296
3.2	Wordpress erweitern	304
3.2.1	Redundanten Code ersetzen	304
3.2.2	Den Inhalt des Posts verändern.....	308
4	Anhang.....	325
4.1	Programmieren mit Stil	325
4.1.1	Code Is Poetry!.....	325
4.1.2	Sei Du selbst!.....	325
4.1.3	Blöcke sind schön!.....	326
4.1.4	80 Zeichen sind genug!.....	328

4.1.5	Das Leerzeichen ist Freund und Feind!	328
4.1.6	Optionen kommen zum Schluss!.....	329
4.1.7	Was gehört wohin?.....	330
4.1.8	Kein Programm ohne Kommentar!	330
4.1.9	Fazit.....	332
4.2	Befehlsreferenz für Wordpress	332
4.2.1	Tags in Wordpress.....	333
4.2.2	Funktionsreferenz	339
4.2.3	Options-Referenz	341
4.2.4	Globale Variablen	343
4.3	Befehlsreferenz für PHP.....	344
4.3.1	Befehle in PHP	344
4.3.2	Parameter zur Formatierung von Ausgaben	347
Stichwortverzeichnis		351

2 Themen in Wordpress

In diesem Kapitel lernen wir *Wordpress* kennen. Wir werden sehen, wie die Software aufgebaut ist, wie sie mit der Datenbank kommuniziert und welche Eigenarten sie hat. Wir werden die dazugehörigen Fachausdrücke und die Terminologie kennenlernen. Danach behandeln wir das Thema-System von *Wordpress*.

Im zweiten Teil des Kapitels werden wir dieses Wissen anwenden und ein eigenes Thema, also eine eigene Gestaltungsvorlage, erstellen. Gleichzeitig erweitern wir unser Wissen um einige Befehle der Programmiersprache PHP. Danach befassen wir uns mit einer bestehenden Vorlage. Hier werden wir uns mit der Erweiterung, der Anpassung und dem Übersetzen von Themen beschäftigen.

Am Ende dieses Kapitels haben wir den Aufbau von *Wordpress* verstanden. Wir kennen die grundlegenden Befehle von PHP und auch die Funktionen und Tags der Blogsoftware. Wir werden in der Lage sein, einfache Anwendungen in PHP selbst zu programmieren. Wir können auch sicher mit der Blogsoftware umgehen und verstehen die einzelnen Module. Zusätzlich werden wir wissen, wann und wie auf die Datenbank zugegriffen wird.

2.1 Der Hintergrund zu Wordpress

Bevor wir uns mit der Programmierung rund um *Wordpress* beschäftigen, müssen wir den Aufbau und die eigene Terminologie kennenlernen. Beginnen wir mit dem Aufbau und der Struktur der Anwendung.

2.1.1 Aufbau

Wir werden in diesem Abschnitt nicht in die Tiefe gehen. Aber ein paar Grundlagen müssen wir trotzdem behandeln und kennen. Die folgende Abbildung bietet einen groben Überblick über die Struktur von *Wordpress* und die Verbindungen der einzelnen Bereiche:

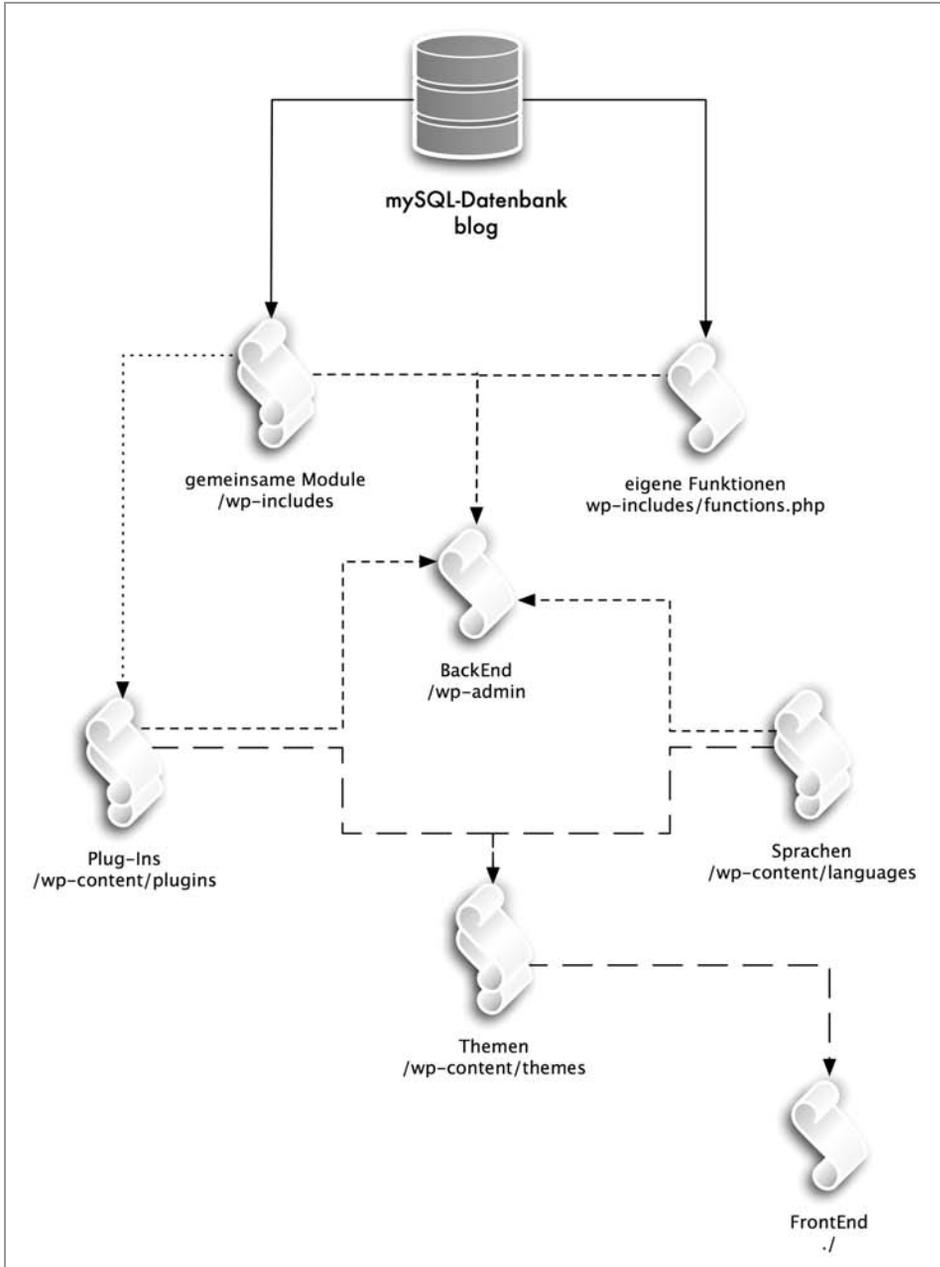


Bild 2.1: Die Struktur von Wordpress

Prinzipiell ist *Wordpress* modular aufgebaut. Dies gibt uns auf der einen Seite die Möglichkeit, die Anwendung zu erweitern, anzupassen und auch in andere Sprachen zu übersetzen. Andererseits ist es anfangs schwierig, die Verbindungen zwischen den einzelnen Modulen und den PHP-Dateien zu verstehen.

Wie wir sehen, besteht die Anwendung aus einer Datenbank, die alle Informationen an das Backend und an das Frontend liefert. Dies geht sogar soweit, dass die Einstellungen der installierten Module, Plug-Ins und Widgets in der Datenbank gespeichert werden.

Die nächsten beiden großen Bereiche sind das Backend und das Frontend. Beide benutzen Module aus den Verzeichnissen */wp-includes* und *wp-content/plugins*.

Im Verzeichnis */wp-includes* sind allgemeine Module, Dateien und Bilder abgelegt. Diese enthalten die grundlegenden Funktionen für die Anwendung und werden in allen anderen Bereichen immer wieder verwendet.

Das Backend dient der Administration des Blogs und ist hauptsächlich im Verzeichnis */wp-admin* abgelegt. Hier werden alle Aufgaben zur Administration des Blogs erledigt. Wir können hier neue Module und Plug-Ins installieren und weitere Themen hinzufügen und aktivieren. Aber auch Kommentare und Artikel, in der Fachsprache *Posts* genannt, können wir hier erstellen und verwalten. Der Administrationsbereich enthält einerseits Daten aus der Datenbank, verwendet aber andererseits die Plug-Ins aus dem Verzeichnis *wp-content/plugins* sowie die oben beschriebenen allgemeinen Funktionen und Module.

Das Frontend ist der Bereich, den der Besucher des Blogs sieht, also das Blog an sich. Dieser Bereich ist im Hauptverzeichnis abgelegt. Es werden aber auch die allgemeinen Funktionen der Anwendung genutzt. Zudem wird das Frontend hauptsächlich von Modulen im Verzeichnis */wp-content* versorgt. Wobei im Verzeichnis */wp-content/plugins* die einzelnen Zusatzmodule, im Verzeichnis */wp-content/languages* die verschiedenen Übersetzungen und im Ordner */wp-content/themes* die verschiedenen Themen abgelegt sind.

2.1.2 Terminologie von Wordpress

Die Blogsoftware hat, wie alles im Bereich der Informationstechnologie, ihre eigene Fachsprache und ihre eigenen Ausdrücke. Wir werden hier nicht alle kennenlernen, aber die wichtigsten für die Anpassung von *Wordpress* müssen wir auf jeden Fall beherrschen.

Fachausdrücke des Inhalts

Hier lernen wir einige Fachwörter kennen, die den zu erstellenden und zu verwaltenden Inhalt betreffen.

Wir sprechen oft von Artikeln oder Beiträgen, diese werden in der Fachsprache *Posts* genannt und bilden den grundlegenden Inhalt eines Blogs. Meistens sind gerade die Posts der Grund, warum ein Blog überhaupt existiert.

Die Kategorien (*categories*) sind die Hauptthemen des Blogs. Die einzelnen Posts werden in Kategorien sortiert, damit der Leser einen besseren Überblick erhält. Zusätzlich kann jeder Post mit mehreren Schlüsselwörtern versehen werden. Diese Wörter werden *Tags* genannt und klassifizieren die Posts zum schnelleren Wiederfinden der behandelten Themen. Diese beiden Ordnungsgruppen werden auch als *Meta*-Daten oder *Meta*information bezeichnet. Sie sind nicht identisch mit dem Inhalt des Posts, sondern beschreiben den Inhalt mit Überbegriffen. Dazu gehören aber auch der Name des Autors und das Erstellungsdatum des Posts.

Sobald ein Artikel veröffentlicht wurde, kann ihn der Leser kommentieren. Diese Kommentare (*comments*) sind der zweite Grund für einen Blog. Dadurch können bzw. sollen Interaktionen zwischen den Autoren und den Lesern entstehen, der Blog lebt sozusagen.

Der letzte Bereich eines Blogs sind die Seiten (*Pages*). Diese sind meist statische Informationen wie »Über mich« oder »Impressum«. Der Inhalt von Seiten ist eher zeitlos und darf nicht mit den Posts verwechselt werden. Seiten können normalerweise vom Anwender kommentiert werden, aber eine Kategorisierung durch den Administrator ist nicht möglich.

Fachausdrücke des Designs

Gerade beim Design zeigt sich die Flexibilität von *Wordpress*. Einer der zentralen Punkte fürs Design ist der sogenannte *Loop*. Verständnis dieses Punktes ist von grundlegender Bedeutung, wenn man sich mit der Entwicklung eines Themas beschäftigt. Daher werden wir ihn weiter unten extra behandeln. Der Loop ist die kritische PHP-Struktur, die für das Anzeigen der Posts zuständig ist.

Das zweite große Thema im Template-Design sind die *Template-Tags*. Dies sind spezielle Funktionen von *Wordpress*, die in PHP programmiert wurden, um Aktionen auszuführen oder Informationen anzuzeigen. Template-Tags sind also Grundbestandteile eines Templates, die die Struktur und den Ablauf der Website kontrollieren. Jedes Template holt sich die notwendigen Informationen aus der *mySQL*-Datenbank und erzeugt eine HTML-Seite, die zum Browser des Lesers gesendet wird.

Die Template-Hierarchie (*template hierarchy*) kontrolliert so gut wie alle Aspekte der Ausgabe. Damit sind nicht nur die Posts und Kommentare gemeint, sondern auch die Kopfdaten (*Header*), die Seitenleisten (*sidebars*) und die Archive. Die Archive (*archives*) stellen allerdings nur eine dynamisch generierte Liste von Posts dar und werden üblicherweise nach Datum, Tag, Kategorie oder Autor zusammengefasst.

Wir haben jetzt schon öfter den Begriff *Template* und auch *Template-Tag* gehört. Dies sind die beiden Bestandteile eines *Wordpress*-Themas (*Theme*). Ein Thema ist das gesamte Design eines Blogs und umfasst die Bereiche Farben, Bilder und Text. Themen werden in eigene Unterverzeichnisse im Ordner */wp-content/themes/* abgelegt. Sie bestehen aus Dateien (*Templates*) für die einzelnen Themenbereiche wie die Seitenleiste, die Seiten oder die Fehlermeldungen. Zusätzlich enthält der Ordner Informationen fürs Layout, und zwar in einer CSS-Datei mit dem Namen *style.css*. Auch können im Unterordner *images* zusätzliche Bilder enthalten sein. Die Flexibilität von *Wordpress* geht so weit, dass der Benutzer mit einem passenden Plug-In sein eigenes Thema für die Website auswählen kann. Damit könnte jeder Leser dieses Buches das Aussehen des Blogs nach seinen Bedürfnissen anpassen.

Plug-Ins sind benutzerdefinierte Funktionen bzw. Module, die die Funktion der Anwendung erweitern. Mit Plug-Ins kann jeder Benutzer zusätzliche Features für *Wordpress* programmieren. Durch das Verwaltungsmodul und das zentrale Plug-In-Verzeichnis auf der Website von *Wordpress* ist es sehr einfach, die Blogsoftware zu erweitern und mit nahezu jeder gewünschten Funktion zu versehen. Auch diese Teile der Anwendung haben ein eigenes Verzeichnis, es befindet sich in */wp-content/plugins/*.

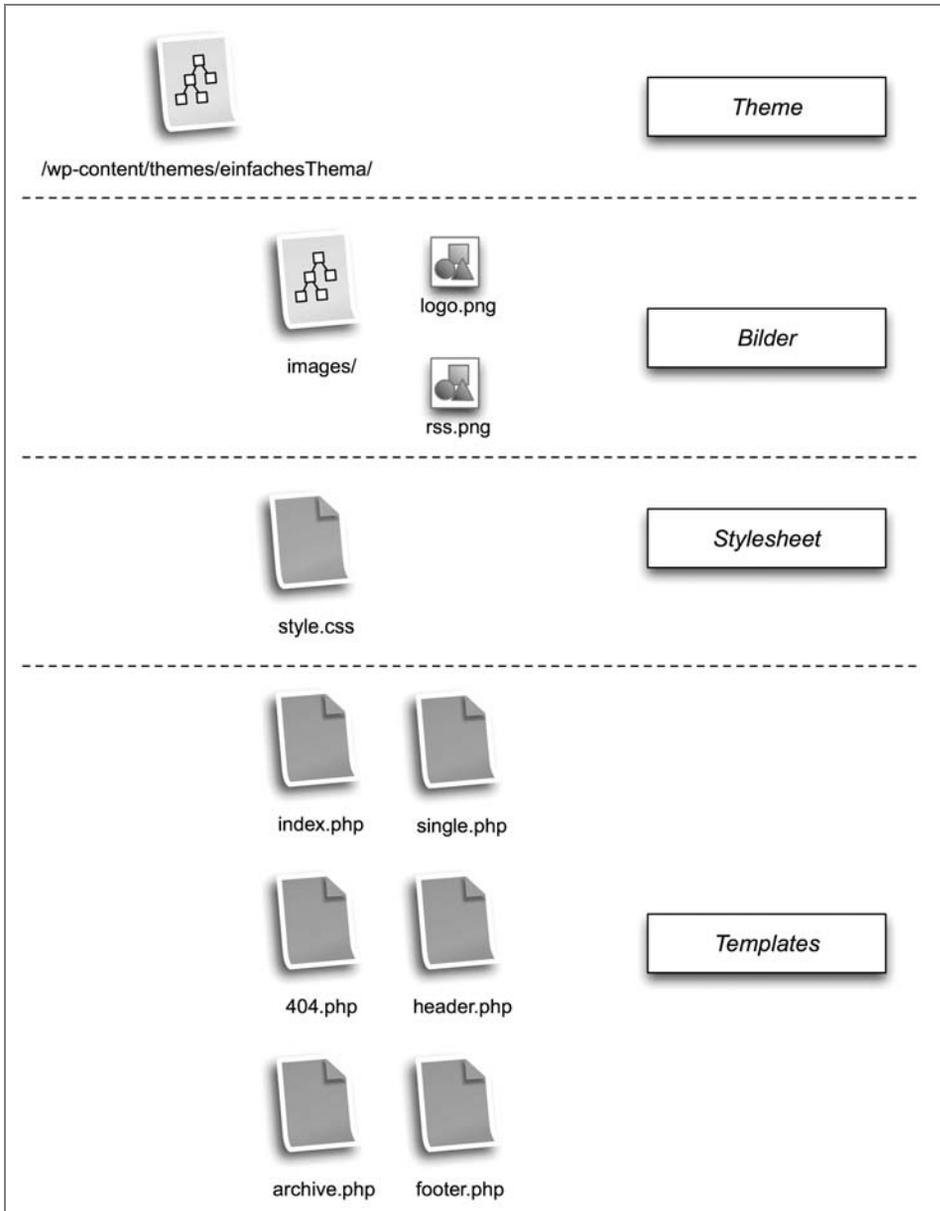


Bild 2.2: Struktur eines Themas in Wordpress

Fachausdrücke für die Entwicklung

Der *Query-String* ist ein Teil des Uniform Resource Identifiers (URI). Damit können Webseiten dynamisch feststellen, welche Daten angezeigt werden sollen. Der Query-String beginnt immer mit einem Fragezeichen und kann verschiedene Parameter enthalten. Diese werden immer mit einem Ampersand (&) getrennt. Der Parameter selbst besteht immer aus seinem Namen, gefolgt von einem Gleichheitszeichen und danach dem Wert selbst. Da Query-Strings oft Suchmaschinen vom Indizieren der Seiten abhalten, wird versucht, diese nicht darzustellen. Auch dafür stellt *Wordpress* eigene Methoden im Backend zur Verfügung, die sogenannten Permalinks, um dies unabhängig vom System des Webservers zu erreichen. Eine komplette URI mit einem Query-String könnte zum Beispiel wie folgt aussehen.

```
http://www.example.com?category=Design&month=07
```

Die *Query-Variable* ist eine einzelne Variable, die mit dem Query-String übergeben wird. Im obigen Beispiel wäre `category` die Query-Variable und `Design` der Wert der Variablen.

2.1.3 Die Hierarchie von Templates

Die Unterschiede zwischen Thema und Template haben wir ja schon kennengelernt. Nun müssen wir uns noch damit beschäftigen, wie Templates zu einem Thema zusammengefügt werden.

Wordpress benutzt generell den Query-String, um zu bestimmen, welches Template bzw. welches Template-Set aufgerufen werden soll. Zuerst überprüft die Anwendung den Query-String und entscheidet, welche Informationen präsentiert werden sollen, sei es eine Suchseite, ein einzelner Post oder eine Seite. Danach werden die Templates in der Folge der Hierarchie und Verfügbarkeit im Thema ausgewählt und der Inhalt erzeugt.

Alle Template-Dateien sind im Ordner des aktiven Themas abgelegt. *Wordpress* durchläuft die in der folgenden Abbildung 2.3 dargestellte Hierarchie und verwendet den Dateinamen für die ERSTE Übereinstimmung in der Liste.

Mit Ausnahme der Datei `index.php` können wir entscheiden, ob ein bestimmtes Template verwendet werden soll oder nicht. Solange *Wordpress* keine passende Template-Datei für einen Eintrag findet, geht es zum nächsten Eintrag in der Hierarchie. Sollte die Anwendung keine einzige Übereinstimmung finden, so wird die Basisdatei `index.php` verwendet.

Diese Abbildung zeigt die komplette Template-Hierarchie. Außerdem werden die in Wordpress verwendeten Funktionen dargestellt, die die Art des Templates überprüfen.

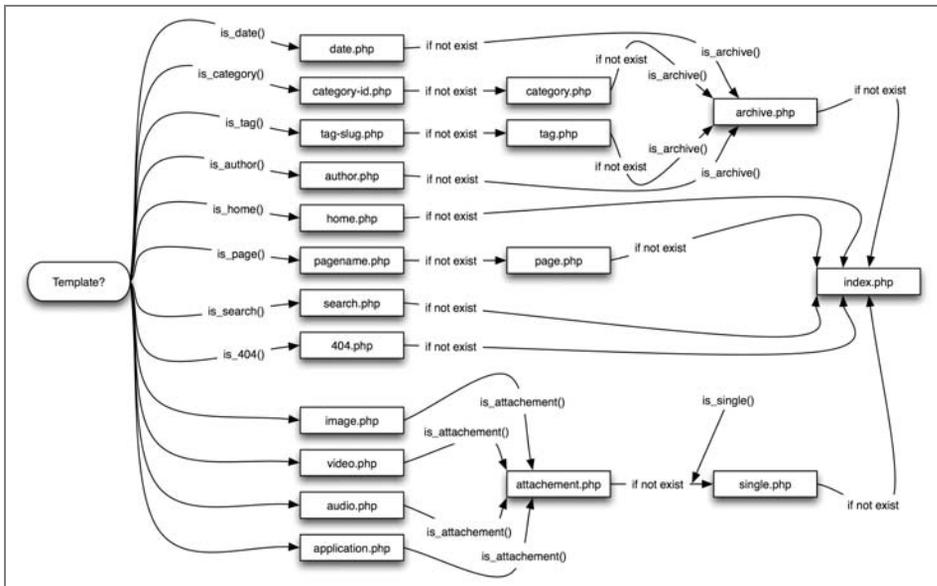


Bild 2.3: Hierarchie der Templates

Gehen wir von folgendem Beispiel aus, um die Hierarchie zu verdeutlichen. In unserem Blog gibt es eine Kategorie *Design* mit der ID 4 und einem Thema mit dem Namen *wpKlar*. Der Anwender will jetzt alle Posts dieser Kategorie sehen, er ruft also die URL <http://www.example.com/category/Design/> auf. *Wordpress* übersetzt *Design* mit Hilfe der Datenbank in die Zahl 4. Als nächstes wird die Hierarchie durchlaufen. Es wird also eine Datei *category-4.php* im Verzeichnis */wp-content/themes/wpKlar/* gesucht. Wird diese gefunden, so wird sie als Vorlage verwendet, mit Inhalt gefüllt und dem Anwender angezeigt. Existiert sie hingegen nicht, sucht *Wordpress* nach einer Datei *category.php* im selben Ordner. Wird die allgemeine Vorlage für die Kategorien gefunden, so wird diese mit Inhalten gefüllt und an den Browser des Lesers gesendet. Wird diese jedoch auch nicht gefunden, so verwendet die Anwendung die generelle Vorlage *index.php*.

2.1.4 Der Loop

Der Loop wird von *Wordpress* zum Anzeigen der einzelnen Artikel bzw. Kommentare verwendet. Die Anwendung verarbeitet jeden einzelnen Artikel, der auf der aktuellen Seite angezeigt werden soll. Der Post wird entsprechend den Tags-Kriterien innerhalb des Loops formatiert. Das bedeutet aber auch, dass jeder HTML- oder PHP-Code innerhalb des Artikels wiederholt dargestellt bzw. ausgeführt wird. Wollen wir beispielsweise ein Inserat am Ende jedes Artikels einblenden, so muss der entsprechende Code dafür innerhalb des Loops geschrieben werden.

Der Loop wird mit PHP als Schleife abgebildet. Der Anfang des Loops sieht wie folgt aus:

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

Nach dem Beginn der Schleife folgt der eigentliche Code, der die Posts verarbeitet und anzeigt. Das Ende schließt die Schleife ab und gibt unter Umständen eine Meldung aus, dass keine Posts gefunden wurden.

```
...  
<?php endwhile; else: ?>  
<p>Leider wurden keine passenden Artikel gefunden.</p>  
<?php endif; ?>  
...
```

Wenn wir unsere eigenen Templates erstellen, sind wir auf den Loop angewiesen. Denn ohne ihn können wir nur jeweils einen Artikel anzeigen.

Der Loop läuft nach einem festgelegten Schema ab. Zuerst überprüft *Wordpress*, ob alle notwendigen Dateien vorhanden sind. Danach werden die Standardeinstellungen aus der Datenbank geholt. Dies sind beispielsweise die Anzahl der anzuzeigenden Posts, ob Kommentare erlaubt sind und Ähnliches. Wenn diese Daten verarbeitet wurden, überprüft *Wordpress* die Anfrage des Benutzers. Damit kann die Anwendung feststellen, welche Artikel von der Datenbank geladen und angezeigt werden sollen.

Fordert der Benutzer keine Kategorie, keinen bestimmten Artikel oder keine Seite an, verwendet die Applikation die am Anfang ermittelten Standardwerte, um festzulegen, welche Posts für den Anwender zusammengestellt werden sollen. Haben wir im Backend festgelegt, dass nur zehn Artikel angezeigt werden sollen, so ermittelt *Wordpress* die zehn aktuellsten Artikel aus der Datenbank.

Sobald diese Schritte erledigt sind, stellt die Anwendung eine Verbindung zur Datenbank her, holt sich die notwendigen Informationen und speichert sie in Variablen. Der Loop greift in der Folge auf diese Variablen zu und verwendet die gespeicherten Werte, um sie in unserem Template anzuzeigen.

Wenn der Leser unseren Blog aufruft, hat er keine speziellen Informationen angefordert. In diesem Fall benutzt *Wordpress* automatisch die Datei *index.php*, um alles anzuzeigen. Am Anfang werden wir uns mit dieser Datei beschäftigen, um den Loop sicher zu beherrschen. Erst danach werden wir uns mit den anderen Templates befassen und verschiedene Tricks kennenlernen.

2.2 Themen

2.2.1 Aufbau eines Themas

Wie wir schon weiter oben erfahren haben, besteht ein Thema aus verschiedenen Dateien, wobei der Großteil Template-Dateien sind. Diese Dateien lassen sich in drei große Gruppen aufteilen:

- Dateien für bestimmte Zustände.
Diese kennen wir bereits aus der Template-Hierarchie. Sie werden je nach Anforderung von *Wordpress* verwendet, um den Inhalt zu präsentieren.
- Dateien mit ausgelagertem Code.
Diese Dateien enthalten Quellcode oder Inhalte, die immer wieder benötigt werden, beispielsweise Stylesheets oder die Dateien *header.php*, *footer.php*, die den Kopf- und Fußbereich des Blogs darstellen oder *functions.php*, welche stets wiederkehrende Programmteile enthält.
- Sonstige Dateien.
Dies sind alle Dateien, welche sich nicht in eine der ersten beiden Gruppen einordnen lassen. Darunter fällt zum Beispiel ein Bild vom Aussehen des Templates, welche die Auswahl im Backend erleichtert.

Welche Dateien den Bereichen im Einzelnen zugeordnet werden, sehen wir in der Folge beim Aufbau unseres ersten Themas. Grundsätzlich lässt sich das Design oder besser die Struktur eines Themas sehr leicht in Sektionen darstellen.

Alle vier Bereiche stehen für eine oder mehrere Template-Dateien. So gibt es für den Kopf- und Fußbereich je eine Datei, nämlich *header.php* und *footer.php*. Für die Seitenleiste (*sidebar*) kann es eine oder mehrere Dateien geben, je nachdem, wie viele Seitenleisten angezeigt werden sollen. Im Inhaltsbereich kommt die Template-Hierarchie zum Tragen. Dies heißt, dass je nach Anforderung verschiedenste Dateien verwendet werden. Auch die Häufigkeit, wie oft diese einzelnen Dateien eingebunden werden, wird durch die Einstellungen im Backend und den Loop bestimmt. Um diese Struktur abbilden zu können, brauchen wir bestimmte Befehle oder Strukturen, welche wir jetzt kennenlernen werden.

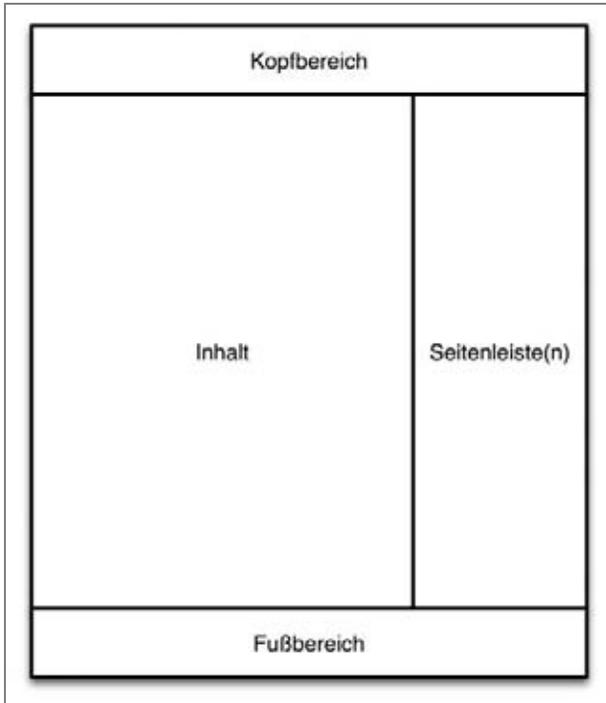


Bild 2.4: Bereiche eines Themas

2.2.2 Include-Tags

Um den gemeinsam verwendeten Code aus den Auslagerungsdateien verwenden zu können, müssen wir ihn natürlich in die einzelnen Template-Dateien einbinden können. Prinzipiell würde es dafür einen eigenen Befehl in PHP geben, welcher aber eine exakte Pfadangabe zur Datei braucht. Dadurch hätten wir nicht die Flexibilität, die wir in *Wordpress* gerne nutzen, denn damit wären die Pfadangaben vorgegeben und könnten nicht durch die Installation selbst bestimmt werden.

Daher gibt es eine eigene Funktion innerhalb der Anwendung, die die richtige Datei aus dem aktiven Thema einbindet. Sie funktioniert prinzipiell gleich wie die Entsprechung in PHP, kennt aber den passenden Dateinamen und den Pfad, um die Datei zu verwenden.

Der Befehl `<?php get_header(); ?>` liefert den Code für den Kopfbereich. Anders gesagt, wird an dieser Stelle die Datei *header.php* eingefügt. Der Fußbereich bzw. die Datei *footer.php* wird mit `<?php get_footer(); ?>` eingebunden. Die Seitenleiste aus der Datei *sidebar.php* fügen wir mit `<?php get_sidebar(); ?>` ein. Den Bereich für die Kommentare eines Posts oder einer Seite, die mit der Datei *comments.php* erzeugt werden, können wir mit `<?php comments_template(); ?>` inkludieren. Zu guter Letzt

können wir dem Leser auch anbieten, unseren Blog zu durchsuchen. Die dafür zuständige Datei ist *searchform.php* und wird mit dem Include-Tag

```
<?php get_search_form(); ?>
```

in das Template eingefügt.

2.3 Ein neues Thema

Mit dem bisher angesammelten Wissen sind wir jetzt in der Lage, unser eigenes neues Thema zu gestalten. Wir werden hierzu die einzelnen Template-Dateien anlegen und gleichzeitig die wichtigsten Befehle von *Wordpress* und PHP kennenlernen.

2.3.1 Ein Projekt in Eclipse anlegen

Um unser Thema zu erzeugen, starten wir unsere Entwicklungsumgebung *Eclipse*. Zuerst schließen wir das Register MY APTANA im Editor. Nun müssen wir ein neues Projekt anlegen. Dazu aktivieren wir die Projektansicht in der linken Seitenleiste, indem wir auf das Register PROJECT klicken. Danach rufen wir das Kontextmenü für die Projektverwaltung auf.

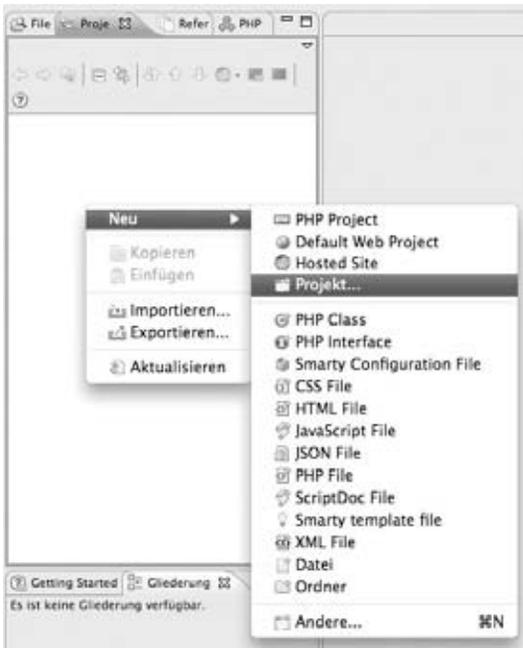


Bild 2.5: Kontextmenü der Projektverwaltung in Eclipse

Mit dem Menübefehl PROJEKT... startet der Assistent, um ein neues Programmierprojekt anzulegen. Aus dem Bereich APTANA PROJECTS wählen wir DEFAULT WEB PROJECT und fahren mit einem Klick auf den Knopf WEITER fort.

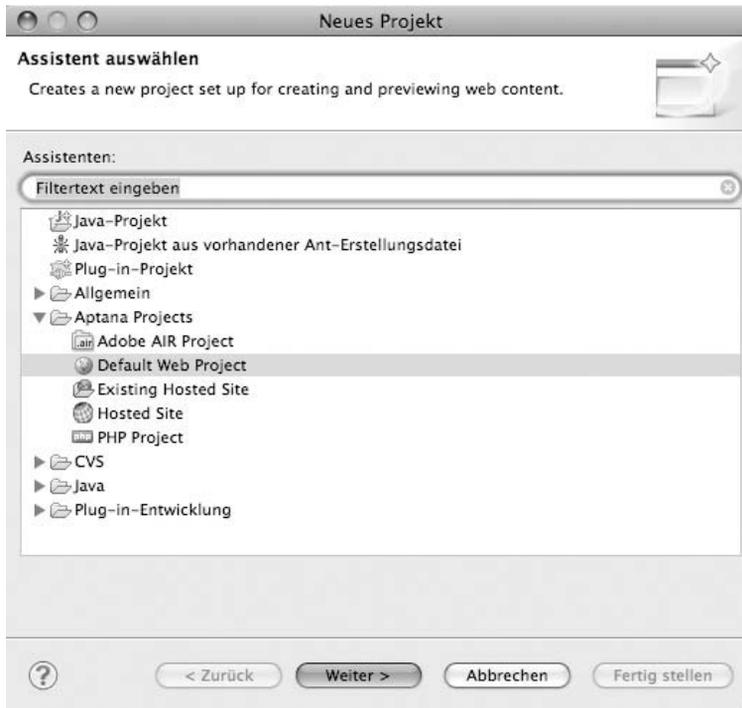


Bild 2.6: Erster Schritt des Assistenten zur Projektanlage

Hinweis

Im Feld FILTERTEXT sollte nichts eingegeben werden! Dadurch werden die einzelnen Projektarten gefiltert und man findet die gewünschte Projektart nicht mehr.

Im nächsten Schritt müssen wir dem Projekt einen Namen geben. Dies ist auch gleichzeitig der Pfadname im Workspace. Da wir bei der Installation den Workspace bereits auf den Basisordner unserer Webserver festgelegt haben, wird dieser automatisch verwendet. Bei der Installation von Wordpress haben wir dieses in einem Verzeichnis */wordpress* im Ordner des Webserver abgelegt. Wenn wir jetzt als PROJEKTNAME ebenfalls *wordpress* verwenden, wird *Eclipse* erkennen, dass der Ordner bereits existiert und die komplette Struktur des Ordners in unser Projekt importieren. Damit haben wir die

gesamte PHP-Anwendung unseres Blogs in einem Projekt zusammengefasst. Mit dem Knopf WEITER kommen wir zum dritten Schritt des Assistenten.

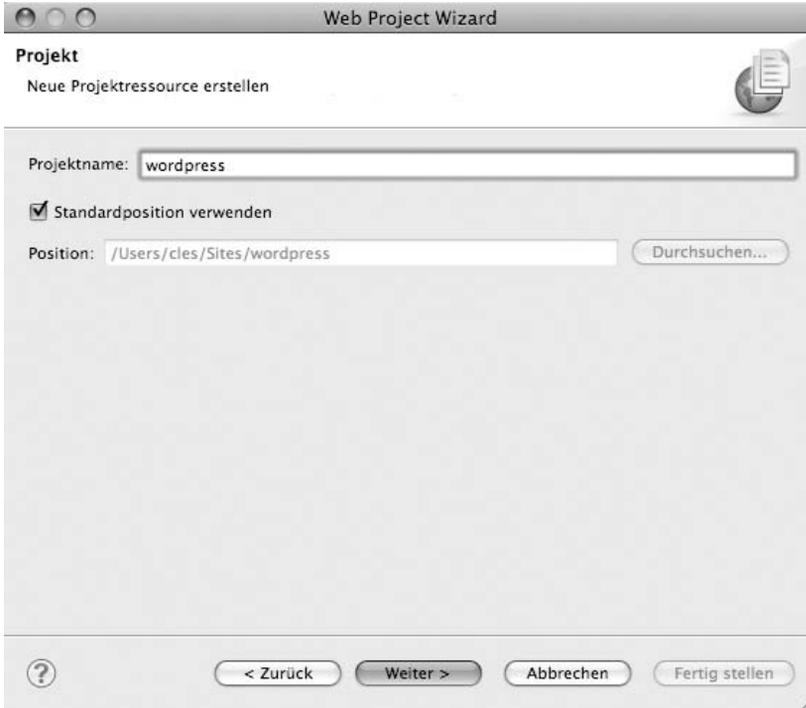


Bild 2.7: Projektname im Assistenten festlegen

Hinweis

Bei manchen Betriebssystemen ist die Unterscheidung zwischen Groß- und Kleinschreibung wichtig. Wenn ein bereits bestehender Ordner verwendet werden soll, muss der Projektname exakt wie der Ordnername geschrieben werden.

Im nächsten Schritt können wir unserem Projekt einige JavaScript-Bibliotheken hinzufügen. Dies ist nicht notwendig, da *Wordpress* bereits alle notwendigen Dateien enthält. Wir können also ohne etwas auszuwählen auf WEITER klicken.

Mit dem letzten Schritt können wir noch eine Verbindung zu einem entfernten Webserver angeben. Dies ist in unserem Fall nicht notwendig, da wir ja mit einer lokalen Installation des Blogs arbeiten. Daher können wir sofort auf FERTIGSTELLEN klicken.



Bild 2.8: Fertigstellen des Assistenten zur Projektanlage

Sobald die Anlage des Projektes abgeschlossen ist, sehen wir in der linken Seitenleiste im Projektexplorer alle Dateien und Unterordner unserer Blogsoftware.

Eclipse hat die Eigenart, bei einem neuen Webprojekt sofort eine Datei *index.html* im Basisverzeichnis des Projektes anzulegen und im Editor zu öffnen. Diese Datei benötigen wir aber nicht. Daher schließen wir den Codeeditor sofort mit einem Klick auf das entsprechende Symbol im Karteikartenreiter.

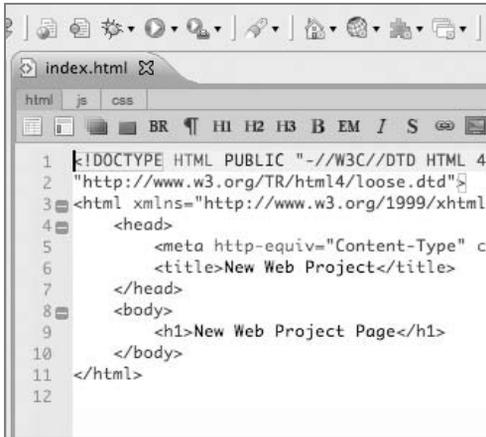


Bild 2.9: Schließen der Datei *index.html*

Nun können wir die Startdatei löschen. Dazu öffnen wir das Kontextmenü für die Datei *index.html* im Projektexplorer und klicken auf den Befehl LÖSCHEN.



Bild 2.10: Kontextmenü zum Löschen von *index.html*

Zur Sicherheit wird uns *Eclipse* nochmals fragen, ob die Datei entfernt werden soll. Dies bestätigen wir mit dem Knopf OK. Wir erhalten so ein Projekt in der Entwicklungsumgebung mit den Originaldateien und der Originalstruktur von *Wordpress*.



Bild 2.11: Sicherheitsabfrage vor dem Löschen einer Ressource

2.3.2 Die Struktur des Themas anlegen

Nun können wir die wichtigsten Dateien und Ordner unseres Themas anlegen. Dazu öffnen wir den Ordner `/wp-content` und den Ordner `/themes` im Projektextplorer von *Eclipse*.



Bild 2.12: Projektextplorer mit geöffnetem Themen-Ordner

Jetzt können wir das Kontextmenü für den Ordner `/themes` öffnen und das Untermenü für den Befehl `NEU` anzeigen. Dort klicken wir auf den Befehl `ORDNER`. Die Entwicklungsumgebung öffnet jetzt ein Fenster, in dem wir sehen, wo der Ordner angelegt wird. Im Datenfeld Ordnername geben wir `prettyNew` ein. Damit haben wir das Verzeichnis und gleichzeitig unser erstes Thema benannt. Mit einem Klick auf den Knopf `FERTIG STELLEN` wird der Ordner im Themenverzeichnis angelegt.



Bild 2.13: Anlegen eines Ordners in Eclipse

Nun benötigen wir für unser erstes Thema vier Dateien:

- *index.php*
- *header.php*
- *footer.php*
- *sidebar.php*

Diese legen wir jetzt in unserem Themenordner an. Dazu öffnen wir das Kontextmenü für den Ordner */prettyNew* und im Untermenü des Befehls NEU klicken wir auf PHP FILE.

Hinweis

Im Kontextmenü stehen sehr viele Arten von PHP-Dateien. Es ist wichtig, dass nur der Befehl PHP FILE verwendet wird.

Eclipse zeigt uns hier ein Fenster mit dem kompletten Pfad unseres Themen-Ordners im Datenfeld CONTAINER. Hier müssen wir nichts verändern, wenn wir das Kontextmenü für den richtigen Ordner aktiviert haben. Falls der Speicherort nicht stimmen sollte, können wir ihn mit dem Knopf BROWSE jederzeit anpassen.

Im Datenfeld FILE NAME geben wir jetzt `index.php` ein und bestätigen den Dialog mit dem Knopf FERTIG STELLEN. Diesen Arbeitsschritt wiederholen wir für jede der oben angeführten PHP-Dateien.

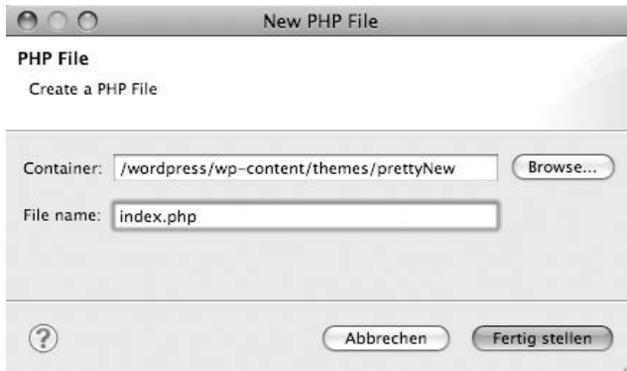


Bild 2.14: Anlegen einer Datei in Eclipse

Da die Entwicklungsumgebung jede neu angelegte Datei sofort öffnet, sollte unsere IDE wie in der folgenden Abbildung aussehen:

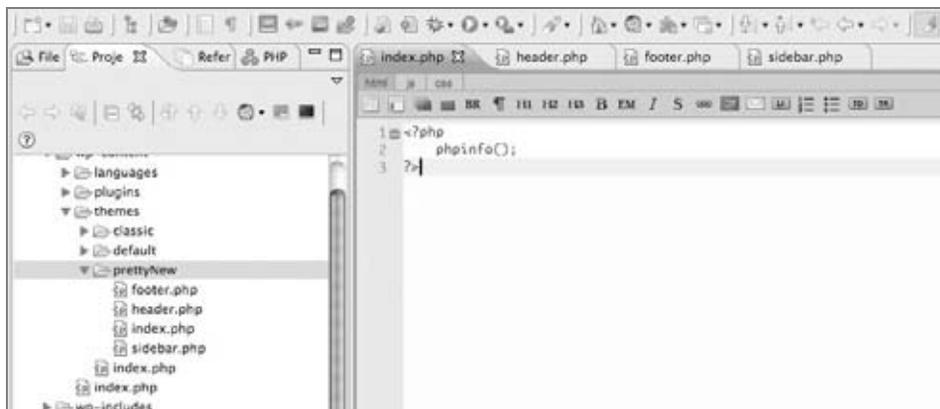


Bild 2.15: Die Grundstruktur des Themas *prettyNew*

2.3.3 Grundlagen der Programmierung

Bevor wir ein neues Thema für *WordPress* programmieren, müssen wir ein paar notwendige Grundlagen der PHP-Programmierung erlernen. Wir werden hier nur die wichtigsten Punkte behandeln und die weiterführenden Techniken bei ihrem ersten Auftreten betrachten.

Wie funktioniert PHP?

Im Grunde gibt es zwei Arten von PHP-Programmen, nämlich reine PHP-Programme und solche mit gemischtem Sourcecode. Der Unterschied ist die Verwendung von HTML und PHP innerhalb derselben Datei. Bei *Wordpress* vermischen sich HTML und PHP, es fällt also in die zweite Kategorie.

PHP-Code kann in den einzelnen Dateien an jeder beliebigen Position eingebaut und auch unterbrochen oder mehrfach verwendet werden. Der Beginn und das Ende des Codes werden mit speziellen Markern gekennzeichnet. So kann der PHP-Interpreter diesen Code vor der Auslieferung des Dokuments an den Browser verarbeiten.

Es gibt mehrere Varianten, diese Kennzeichnungen für den Interpreter zu setzen. Die Kurzvariante, auch SGML-Stil (Standard Generalised Markup Language) oder Short-Tag genannt, ist eine der Möglichkeiten.

```
<?
    /* hier steht PHP-Code */
?>
```

Die gebräuchlichste Variante verwendet den XML-Stil (Extensible Markup Language). Dadurch ist beim Lesen des Codes sofort klar, welche Sprache verwendet wird. Die Unterscheidung zum Short-Tag ist nur das Kürzel `php` beim öffnenden Marker.

```
<?php
    /* hier steht PHP-Code */
?>
```

Diese Art der Einbindung von PHP-Code wird auch von *Wordpress* verwendet. Daher werden auch wir mit dem XML-Stil arbeiten. Um ein besseres Verständnis für die Arbeit einer PHP-Anwendung zu haben, schauen wir uns die folgende Abbildung an:

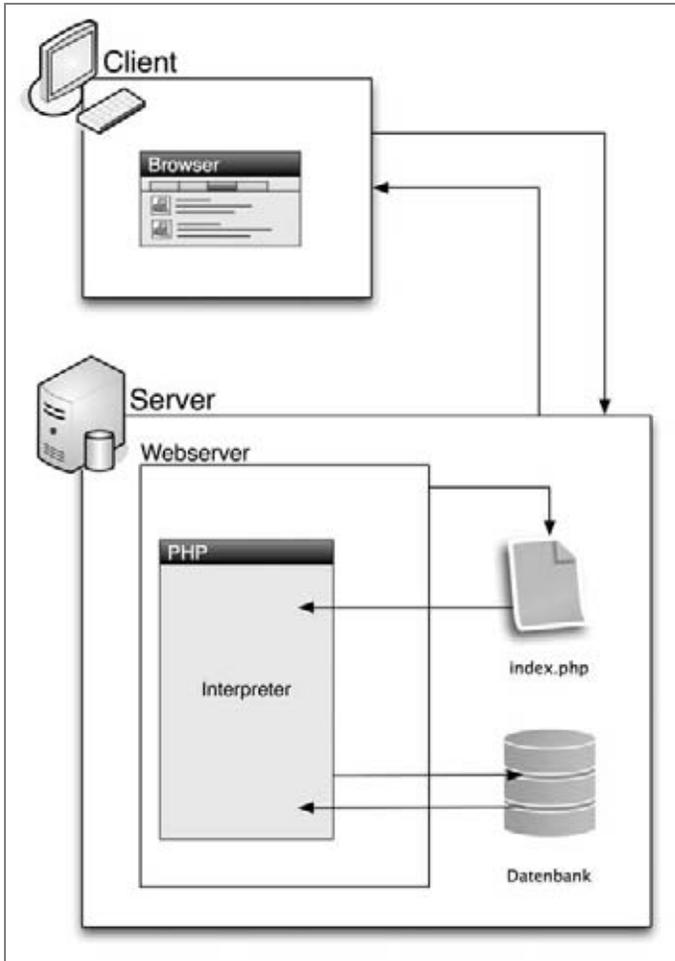


Bild 2.16:
Funktionsweise eines
PHP-Interpreters

Wenn wir die Abbildung betrachten, sehen wir den Ablauf beim Aufruf unseres Blogs. Der Anwender fordert die Datei *index.php* vom Webserver an. Dieser verarbeitet mit Hilfe des PHP-Interpreters den Code innerhalb der Datei und holt zusätzlich die angeforderten Daten, die Posts, Kommentare und Weiteres von der Datenbank. Danach vermischt der Webserver die vorhandenen HTML-Einträge der *index.php* mit dem Resultat vom PHP-Interpreter und liefert dies als HTML-Datei an den Browser des Anwenders aus.

Diese PHP-Skripte erzeugen Ausgaben, die direkt im Browser angezeigt werden. Die gesamte Verarbeitung des PHP-Codes erfolgt also im Webserver. Der Anwender merkt davon so gut wie nichts.

Stichwortverzeichnis

4

404.php 254, 260, 268

A

Ampersand 73

Apache 10

Aptana Studio

erweitern 57

installieren 53

jQuery 57

PHP 57

Prototype 57

Scriptaculous 57

Archive 70

archive.php 240, 241, 247, 268, 277, 308

Array 148

assoziativ 150, 256

Element 150

erstellen 149

Hash 150

numerisch 149

a-Tag 220, 296, 303

B

Backend 69

Bang-Operator 165

Benutzerkontensteuerung 14, 15, 16, 56

Blockoperator 123

Blog durchsuchen 206

body 145

C

Callback 311

CamelCase 89

Categories 70

Comments 70

comments.php 77, 169, 175, 197

Conditional-Tag 252

CSS

a 108

archive 253

body 106

content 219

Datei 106

h1 108

h2 108

Kommentare 106

li 112

p 109

post 219

postContent 220

searchHead 212

sidebar 240

sidebar-page 240

sidebars 112, 237

Suchformular 115

tags 307

theLoop 219

ul 112

Wrapper 107

D

Datenbank	
installieren	40
de_DE.po	294
Dekrement	129
div-Tag	202
Document Type Definition	117

E

Eclipse	50
Aptana Studio installieren	53
Arbeitsbereich festlegen	52
Galileo	50
installieren	51
Perspektive	58
Projekt anlegen	78
Sprachpaket	50
Sprachpaket installieren	51
Workspace	79
Endlosschleife	128
Entwicklungsumgebung	
installieren	49

F

Firefox	60
FireBug	61
GridFox	63
WebDeveloper	62
footer.php	76, 77, 84, 105, 119, 148, 281
footers.php	76
form-Tag	179, 207
Frontend	69
functions.php	166, 179, 225, 267, 281, 295, 306, 309, 311, 312
Funktionen	98

G

globale Variable	209
Gravatar	191

H

h2-Tag	302
head_tag	141
Header	70
header.php	76, 77, 84, 103, 117, 120, 147, 148, 281, 299, 302
Hooks	310, 312
HTML.....	148
a-Tag	220, 296, 303
body	118
body-Tag	145
div-Tag	202
DTD	117
Entität	182
form-Tag	179, 207
h2-Tag	302
h3	131
head-Tag	118, 141
html-Tag	148
input-Tag	207
Kommentare	104
label-Tag	181
li-Tag	296, 303
p-Tag	318, 319
theLoop	122
ul-Tag	151, 191, 194, 296

I

IDE	
installieren	49
Include-Tags	77
index.html	81
index.php	75, 84, 126, 148, 168, 201, 204, 227, 307, 308
Inkrement	129
input-Tag	207

- J**
- Java 50
 - installieren 51
 - Runtime Environment 50
- K**
- Kategorien 70
 - Kennwort
 - Benutzer root 28
 - Kommentar
 - CSS 106
 - Kommentare
 - HTML 104
 - PHP 102
- L**
- label-Tag 181
 - li-Tag 296, 303
 - logische Operatoren 95
 - Loop 70, 74, 126, 188, 194, 201,
 - 207, 212, 216, 228, 232,
 - 241, 281, 285, 289
 - manuell 256
- M**
- Metadaten 70
 - Metainformationen 131, 248, 289, 297
 - mySQL 10
 - installieren 31
- N**
- Namensraum 266
- O**
- objektorientierte Programmierung 210
 - onBlur 207
 - onFocus 207
 - OOP 210
 - \$wp_query 211, 249
 - Class 210
 - Events 210
 - inherit 211
 - Methods 210
 - Objects 210
 - Properties 210
 - WP_Query 211, 252
 - Operator
 - Bang 165
- P**
- Page 70
 - page.php 215, 283
 - Permalink 73
 - Perspektive
 - Aptana 59
 - PHP
 - aktivieren 23
 - Array 90, 148
 - array() 149
 - array_key_exists() 318
 - Ausgabe 91
 - Bang-Operator 165
 - Blockoperator 123
 - bool 90
 - break 270
 - case 270
 - count() 196
 - date() 144
 - define() 267
 - die() 258
 - echo 91
 - else 125
 - elseif 125
 - Entscheidung 122
 - exit() 258
 - explode() 318
 - float 90
 - foreach() 188
 - function 99
 - Funktion 98

- if 123
 - integer 90
 - Interpreter 87
 - intval() 321
 - Kommentarblock 101
 - Kommentare 102
 - Konfigurationsdatei 26, 32
 - logische Operatoren 95
 - Loop 122
 - NULL 90
 - object 90
 - printf() 173
 - Punkt-Operator 92
 - return 100
 - Schleifen 75, 127, 188
 - Short-Tag 86
 - sizeof() 196
 - sprintf() 275
 - str_replace() 319
 - string 90
 - strtotime() 261
 - substr() 246
 - switch() 270
 - Syntax 88
 - testen 20, 28
 - time 144
 - var_dump() 257
 - Variable 88
 - Vergleichsoperatoren 93
 - while 128
 - Zeichenketten verketteten 92
 - Zuweisungsoperator 90
 - phpMyAdmin 31
 - Datenbank installieren 41
 - installieren 33
 - konfigurieren 32
 - testen 36
 - PoEdit 291, 303
 - Post 70
 - Programmierung
 - Class 210
 - Events 210
 - Inherit 211
 - Methods 210
 - Objects 210
 - objektorientiert 210
 - properties 210
 - prozedural 210
 - prozedurale Programmierung 210
 - p-Tag 318, 319
 - Punkt-Operator 92
- Q**
- Query-String 73, 248
 - Query-Variable 73
- S**
- Schleifen 75, 127, 188
 - foreach() 257
 - search.php 207, 284, 308
 - searchform.php 78, 206, 287
 - Seitenleiste 164
 - dynamisch 164
 - SGML 86
 - Short-Tag 86
 - Sidebar 70
 - dynamisch 164
 - sidebar.php . 77, 84, 104, 151, 164, 221, 289
 - single.php 201, 205, 214, 231, 289, 308
 - Standard Generalised Markup
 - Language 86
 - style.css 71, 106, 214, 217, 220,
 - 230, 233, 237, 238, 253, 261, 296
 - Suchen 206
- T**
- Tag 70
 - Tag
 - a 220, 296, 303
 - body 145

div 202
 form 179, 207
 h2 302
 head 141
 html 148
 Include 77
 input 207
 label 181
 li 296, 303
 p 318, 319
 Template-Tag 70
 ul 151, 191, 194, 296
 Template 71
 Kommentare 168
 Template Hierarchy 70, 73
 Template-Tags 70
 TextWrangler 11
 installieren 21
 Thema 71, 76
 Aufbau 76
 Dateigruppen 76
 index.php 101
 neu anlegen 78
 Startdatei 101
 Stylesheet festlegen 106
 Verzeichnis anlegen 83
 Theme 71
 Themenverzeichnis anlegen 83
 Timestamp 261

U

ul-Tag 151, 191, 194, 296
 Uniform Resource Identifier 73
 unitärer
 Dekrement 129
 Inkrement 129
 UNIX-Timestamp 261
 URI 73
 Usability 184, 206, 222, 251
 UTF-8 142

V

Variable 88
 global 209
 Vergleichsoperatoren 93

W

Webserver 11
 aktivieren 22
 testen 19, 22
 Widgets 163, 207
 Windows
 Sicherheitsrichtlinien 15
 Wordpress
 __() 266
 _e() 266
 add_filter() 309
 Administratorpasswort 48
 Archiv 70, 240
 BackEnd 69
 Benutzername 45
 Benutzerrechte 38
 bloginfo() 118
 body_class() 146
 comment_author() 182
 comment_author_email() 184
 comment_author_link() 192
 comment_author_url 185
 comment_date() 193
 comment_id() 191
 comment_text() 192
 comment_time() 193
 comment_type() 192
 comments_open() 170
 comments_popup_link() 137
 comments_popup_script() 138
 comments_template() 169
 Conditional-Tag 222, 252
 Core 103
 Datenbank installieren 40
 do_action() 185

- dynamic_sidebar() 166, 225
- edit_comment_link() 193
- edit_post_link() 134
- einrichten 42
- einzelner Artikel 201, 231
- esc_attr() 181
- esc_attr_e() 181
- FrontEnd 69
- function_exists() 145
- get_avatar() 191
- get_bloginfo() 288
- get_footer() 105
- get_header() 102, 258
- get_option() 171
- get_pages() 299
- get_permalink() 174
- get_post_meta() 314
- get_query_var() 252
- get_search_form() 127
- get_sidebar() 104
- get_the_ID() 314
- get_the_time() 245
- get_userdatabylogin() 248
- has_tag() 251
- have_comments() 187
- have_posts() 130
- Header 70
- Include-Tag 104
- Include-Tags 77
- installieren 37
- is_author() 246
- is_category() 244
- is_date() 252
- is_day() 245
- is_month() 245
- is_page() 222
- is_search() 317
- is_single() 222
- is_tag() 244
- is_user_logged_in() 171
- is_year() 246
- Kategorien 70
- Kommentare 70
- Konfigurationsdatei 42
- Kopfdaten 70
- load_theme_textdomain() 267
- Loop 70, 74, 103, 188, 194, 201,
..... 207, 212, 216, 228, 232,
..... 241, 281, 285, 289
- Metadaten 70
- more-Tag 135
- next_comments_link() 197
- next_post_link() 203
- next_posts_link() 230
- Permalinks 48
- Plug-Ins 71
- Post 69, 70
- post_class() 250
- post_password_required() 170
- posts_nav_link() 138
- previous_comments_link() 196
- previous_post_link() 202
- previous_posts_link() 230
- register_sidebar() 167, 226
- Seite 70
- Seitenleiste 70
- single_tag_title() 245
- split() 247
- Struktur 67
- style.css 106
- Tag 70
- Template 71
 - Hierarchie 70, 73
- Template-Tags 70
- Terminologie 69
- the_author_posts_link() 251
- the_ator() 133
- the_category() 132
- the_content() 135
- the_date() 133

the_excerpt()	213	wp_login_url()	175
the_ID()	204	wp_logout_url()	180
the_permalink()	131	wp_title	142
the_post()	130	Zusatzfunktionen	166
the_tags()	132	Workspace	79
the_time()	133	WordPress	
the_title()	131	single_cat_title()	244
Thema	71	wp-config.php	267, 291
Thema aktivieren	121	X	
Widgets	114, 163	XAMPP	10
wp_count_posts()	228	Control Panel	13, 18
wp_get_recent_posts()	256	Dienste	14, 18
wp_head()	145	installieren	11, 14
wp_link_pages()	136	Z	
wp_list_bookmarks()	159	Zeichencodierung	142
wp_list_categories()	156	Zuweisungsoperator	90
wp_list_comments()	194		
wp_list_pages()	152		

Widmung

Für Florence, die immer Verständnis für mich und meine Ideen hat!

PHP für WordPress

Themes und Templates selbst entwickeln

Weltweit basieren über 200 Millionen Websites auf WordPress und es gibt Tausende von Themes und Templates zum Download. Und dennoch: Wer genaue Vorstellungen von Optik und Funktionalität seiner WordPress-Seite hat, wird selten fündig. Autor Clemens Gull zeigt, wie Sie Ihre Webseiten individuell erweitern: Ein neues Layout, mehrsprachige Seiten, eine zusätzliche Bedienleiste sowie komfortable Archiv- und Kommentarfunktionen sind kein Problem. Der Schlüssel hierfür liegt in der Programmiersprache PHP, in der WordPress entwickelt wurde. Dieses Buch vermittelt Ihnen das nötige PHP-Wissen, damit Ihre WordPress-Seite genau das tut, was Sie wollen.

► **WordPress verstehen und umbauen**

Web-Guru Clemens Gull zeigt, wie es hinter den Kulissen von WordPress aussieht, wie der berühmte WordPress-Loop aufgebaut ist und wo Sie selbst Hand anlegen können. Auch die Struktur von Designvorlagen (Themes) und Templates kommt nicht zu kurz. Dieses Know-how brauchen Sie, um das Seitenlayout zu verändern und zusätzliche Bedienelemente einzufügen. Schritt für Schritt entsteht so ein eigenes WordPress-Theme.

► **PHP-Programmierung für WordPress**

WordPress basiert auf der Skriptsprache PHP. Dieses Buch vermittelt Ihnen genau die PHP-Kenntnisse, die Sie brauchen, um WordPress an Ihre eigenen Vorstellungen anzupassen. Grundbegriffe der Programmierung wie Variablen, Strings, Schleifen, Arrays und Kontrollstrukturen sind bereits nach kurzer Zeit keine Fremdworte mehr. Der Clou: Sie können dieses Wissen gleich dazu nutzen, WordPress umzugestalten, eine ausführliche Befehlsreferenz für PHP- und WordPress-typische Befehle hilft Ihnen dabei.

► **Entwicklertipps aus erster Hand**

Als erfahrener Programmierer weiß Clemens Gull, wo es bei der Softwareentwicklung zu Fehlern und Problemen kommt. Er spart nicht mit Tipps, die Ihnen helfen, Ihr Programm sauber und klar zu gliedern, damit Sie auch später noch nachvollziehen können, was Ihr Code eigentlich macht. Auf diese Weise kommen Sie schnell zu Erfolgserlebnissen.

Aus dem Inhalt:

- Installation von WordPress
- Der Aufbau von WordPress
- Die Entwicklungsumgebung Eclipse
- Nützliche Programmierertools
- WordPress-Themes und -Templates
- Die Template-Hierarchie
- Der WordPress-Loop
- PHP-Grundlagen
- Variablen und Arrays
- Templates für Artikel, Kommentare und Bedienelemente
- Den Blog durchsuchen
- Stylesheets abändern und verbessern
- Das Blog-Archiv
- Programmierung der Fehlerseite
- Mehrsprachige WordPress-Themes
- Befehlsreferenz für WordPress und PHP
- Programmiertipps

Über den Autor:

Clemens Gull studierte Informationstechnologie und Systemmanagement. Er arbeitete als Programmierer und Netzwerkadministrator unter anderem für die Salzburger Sparkasse. Heute leitet er das Webdesignunternehmen Byte Brothers, darüber hinaus ist Gull als Dozent für die Fachhochschule Salzburg und andere Institute aktiv. Sein Weblog „Guru 2.0“ (www.guru-20.info) zählt zu den meistgelesenen deutschsprachigen Blogs zum Thema Internetprogrammierung.



Auf www.buch.cd:

- Webserven und Datenbank für die lokale WordPress-Installation
- WordPress-Installationspaket
- WordPress-Theme mit allen im Buch besprochenen Beispiellistings



30,- EUR [D]

ISBN 978-3-645-60011-8

Besuchen Sie unsere Website

www.franzis.de