WordPress für Fortgeschrittene

Einrichten, anpassen und entwickeln

STRATO

WordPress

STRATO AG
Pascalstraße 10
10587 Berlin
Berlin 2017
www.strato.de

Autoren
Christian Lingnau
Franz Neumeier

Redaktion
Lisa Kopelmann
Oliver Meiners
Michael Poguntke

Satz
Katrin Krause

2. Auflage © 2021 STRATO AG, Berlin

Inhaltsverzeichnis

Einleitung 4

1 Voraussetzungen 5
1.1 WordPress bei STRATO manuell installieren 5
1.2 So nimmst Du Deinen Blog zu STRATO mit 10
1.3 Lokale Testumgebung erstellen 17

2 Layout 23
2.1 Child-Themes: Individuelles WordPress-Design ganz einfach 23
2.2 5 Schönheitstipps für den Kommentarbereich 31
2.3 Autor-Template erstellen 36
2.4 Benutzerdefinierte Felder anlegen 41
2.5 404-Fehlerseite individuell gestalten 45
2.6 Landing Page mit einem Seiten-Template erstellen 50
2.7 Eigenes Theme entwickeln: Bootstrap-Navigation in WordPress implementieren 57
2.8 Eigenes Theme entwickeln: Icons integrieren und Kommentarbereich optimieren 61
2.9 Eigenes Theme entwickeln: Strukturieren mit HTML und CSS 71

3 Plugins 78
3.1 Dein eigenes WordPress-Plugin leicht gemacht 78
3.2 Plugins ersetzen durch Code am Beispiel »Related Posts« 83

4 Backend 87
4.1 WordPress-Login bunter machen 87
4.2 Backend anpassen: Individuell statt von der Stange 91

5 Sicherheit 96
5.1 Strategischer Hintergrund: Welche Sicherheitslücken solltest Du schließen? 96
5.2 Potenzielle Sicherheitslücken von WordPress schließen 100
5.3 Typische WordPress-Fehler beheben 104
5.4 Plugins als Sicherheitsrisiko 114

6 Performance 115
6.1 WordPress Beine machen per .htaccess 115

7 Tracking 119
7.1 Mehr Sichtbarkeit mit strukturierten Daten 119
7.2 An diesen Zahlen erkennst Du, ob Dein Blog erfolgreich ist 123
7.3 Content-Gruppen: Traffic detailliert aufschlüsseln mit WordPress und Google Analytics 130

8 WordPress Multisite 141
8.1 Multisite mit Verzeichnissen und Subdomains 141
8.2 Domain Mapping mit WordPress Multisite 145
8.3 WordPress mehrsprachig machen 148

Schlusswort 151

Einleitung

Liebe Bloggerin, lieber Blogger,

ein eigener Reise-Blog, ein Unternehmens-Blog oder ein Online-Auftritt für das ganz private Projekt – mit WordPress alles kein Problem.

Vielleicht kennst Du schon unser E-Book **›WordPress für Einsteiger** und hast bereits eine eigene Online-Präsenz mit WordPress erstellt. Umso besser! Unser zweites E-Book »WordPress für Fortgeschrittene« zeigt Dir, wie Du noch mehr aus Deinem Blog herausholen kannst.

Du bist noch nicht mit einem passenden Paket ausgestattet?
›Wähle hier den WordPress-Tarif, der zu Deinen Anforderungen passt.

Unsere Autoren Christian Lingnau und Franz Neumeier geben Dir Experten-Tipps dazu an die Hand, wie Du mit Deinem WordPress startest, Deinem Blog ein einzigartiges Aussehen verleihst, eigene Plugins erstellst und typische Fehler behebst. Sie erklären Dir außerdem, welche Sicherheitslücken Du beachten solltest, wie Du sie beheben kannst und woran Du erkennst, ob Dein Blog erfolgreich ist.

Alle Texte und Beispiele basieren auf unserem **›Hosting-Paket WordPress Plus**.

Wir wünschen Dir viel Spaß beim Lesen und Bloggen!

Dein STRATO Team

1 Voraussetzungen

1.1 WordPress bei STRATO manuell installieren

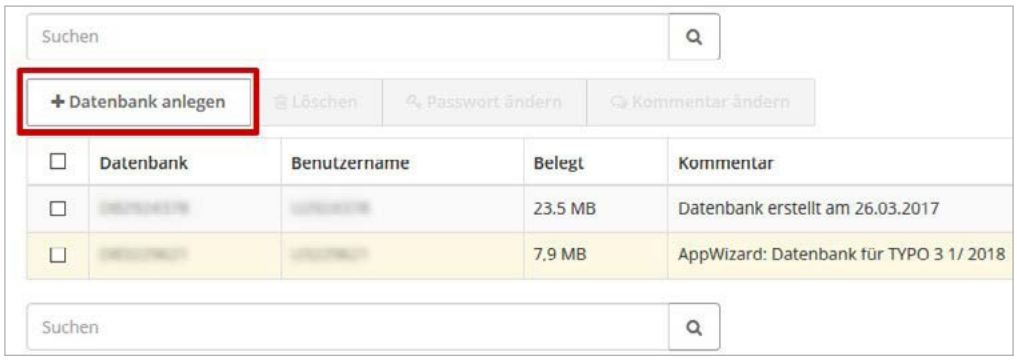
WordPress installierst Du bei STRATO Hosting-Paketen über WordPress & Co. mit einem Mausklick. Willst Du allerdings mehr Kontrolle über die WordPress-Installation, richtest Du WordPress mit dieser Anleitung unkompliziert von Hand ein.

Die manuelle Installation von WordPress verläuft in drei Schritten: Datenbank anlegen, WordPress vorbereiten sowie die Installation im Webspace inklusive der Anpassung einiger Optionen durchführen.

1. Datenbank anlegen

Sofern Du in Deinem Hosting-Account über genügend Datenbanken verfügst, empfiehlt es sich, für WordPress eine eigene Datenbank zu verwenden, statt sie mit anderen Anwendungen zu teilen. Du kannst aber auch mit einer bereits vorhandenen Datenbank arbeiten, denn WordPress nutzt für die Tabellen standardmäßig das Präfix **wp_** und vermeidet so Kollisionen mit anderen Anwendungen.

Um eine neue Datenbank anzulegen, klickst Du einfach auf den Button **Datenbank anlegen** in Deinem STATO Kunden-Login unter **Datenbanken und Webspace > Datenbankverwaltung**.



Die Informationen zur neu angelegten Datenbank werden nun angezeigt. Notiere Dir Datenbank- und Benutzernamen für die spätere WordPress-Installation.

2. Datenbank-Passwort ändern

Ändere nun das Passwort der Datenbank und verwende dabei eine möglichst lange Kette aus Groß- und Kleinbuchstaben sowie Ziffern. Notiere das neue Passwort ebenfalls für später und leg es gut ab. Du benötigst es zu einem anderen Zeitpunkt noch mal. Am besten schreibst Du das Passwort in einem Text-Editor und kopierst es in das Formular.
So vermeidest Du auch bei komplizierten – und damit sicheren – **›Passwörtern** Tippfehler.

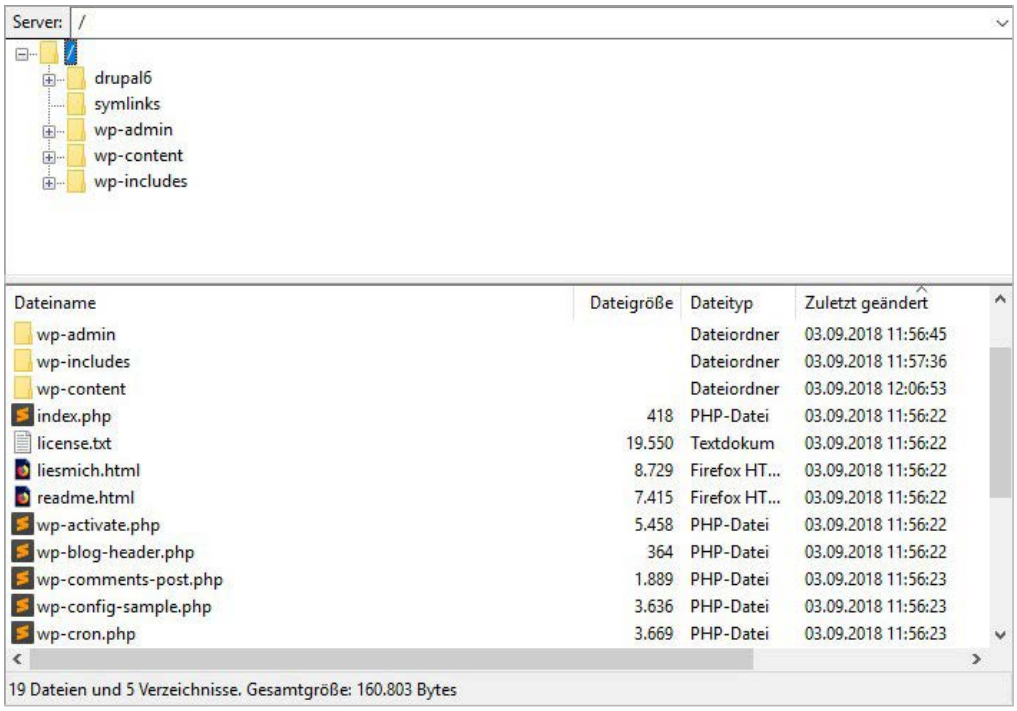
In der Übersicht kannst Du auch den Kommentar zur Datenbank ändern, beispielsweise in »wordpress«, damit Du später weißt, welche Datenbank zu Deiner WordPress-Installation gehört.

3. WordPress herunterladen

Jetzt lädst Du die aktuellste Version von WordPress herunter: **›https://de.wordpress.org/download/**. Entpacke die heruntergeladene ZIP-Datei auf Deiner lokalen Festplatte. Kontrolliere, ob dabei die Verzeichnisstruktur des Unterordners »wordpress« komplett erhalten geblieben ist. Sie enthält die Verzeichnisse **wp-admin**, **wp-includes** und **wp-content** sowie einige PHP-Dateien.

4. WordPress hochladen

Lade nun den Inhalt des Verzeichnisses »wordpress« in Deinen Webspace. Am besten verwendest Du dazu den **›SFTP-Zugang**. Die Verzeichnisstruktur sieht danach so aus:



5. WordPress installieren

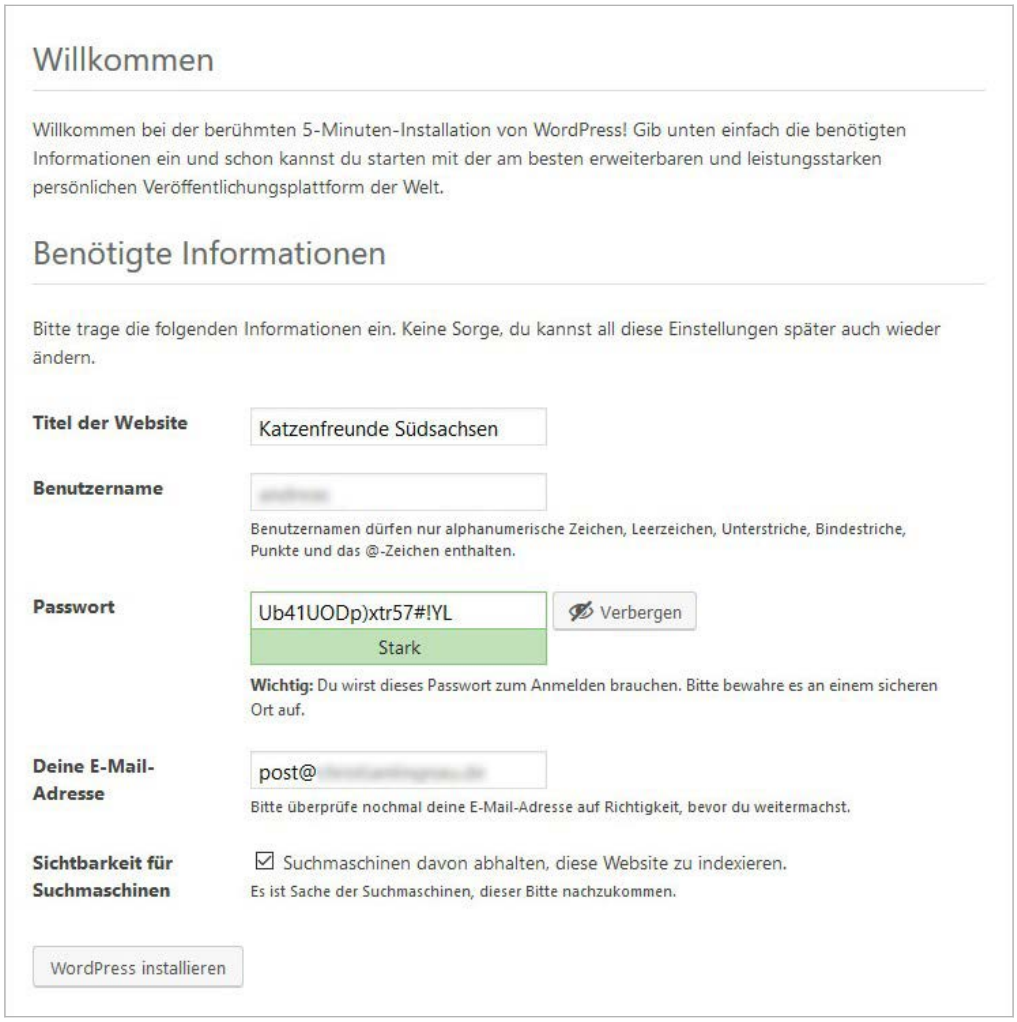
Wenn Du nun das WordPress-Unterverzeichnis Deiner Domain aufrufst (z. B. https://katzenfreunde-suedsachsen.de/wordpress), startet die Installationsroutine. Gib im zweiten Dialog den zuvor notierten Datenbanknamen, Benutzernamen und das Passwort ein. Als Datenbank-Host trägst Du »rdbms.strato.de« (statt »localhost«) ein.

Ändere außerdem das standardmäßige Tabellen-Präfix: Statt **wp_** könntest Du beispielsweise Deine Initialen oder eine andere, kurze und willkürliche Buchstabenfolge verwenden. Das bietet zusätzlichen Schutz vor Hackern.

Sobald die Verbindung zur Datenbank hergestellt ist, musst Du noch ein paar grundlegende Daten eingeben, bevor die eigentliche Installation startet: den Namen Deines Blogs, einen Benutzernamen und ein Passwort (beides notieren!) sowie Deine E-Mail-Adresse. Der Benutzername, den Du hier anlegst, bekommt Administrations-Rechte. Wähle daher einen Benutzernamen, der nicht ohne Weiteres für Dritte erkennbar ist und keinesfalls »admin« oder »Administrator« lautet.

Unter »Sichtbarkeit für Suchmaschinen« entscheidest Du, ob Suchmaschinen Dein Blog indexieren dürfen. Obwohl Du das in den meisten Fällen natürlich willst, solltest Du hier zunächst ein Häkchen setzen und Suchmaschinen so davon abhalten, Deine Website zu listen. Erst nachdem Du Deinen Blog fertig eingerichtet hast (u. a. Impressum und Datenschutz-Hinweise), solltest Du das Indexieren erlauben.

Denke daran, das Häkchen später wieder zu entfernen, denn sonst sperrst Du Google und andere Suchmaschinen dauerhaft aus. Um diese Option zu ändern, gehst Du in der WordPress-Admin-Oberfläche in den Bereich **Einstellungen > Lesen**.



Abschließend klickst Du auf **WordPress installieren**. Danach kannst Du Dich mit Deinen zuvor angelegten Zugangsdaten einloggen.

1.2 So nimmst Du Deinen Blog zu STRATO mit

Einen WordPress-Blog von einem anderen Web-Hoster umzuziehen, ist mithilfe des Plugins **›Duplicator** bequem zu bewältigen. Mit ein paar Tricks und einer Schritt-für-Schritt-Anleitung gelingt Dir der Umzug ganz leicht.

Der Umzug eines WordPress-Blogs ist eigentlich keine große Sache: Datenbank und Dateien kopieren, ein paar Konfigurations-Dateien anpassen, fertig. Aber wie so oft steckt der Teufel im Detail. Mit dem Plugin Duplicator gehst Du den meisten potenziellen Problemen aus dem Weg. Das Plugin kümmert sich automatisch um die richtige Konfiguration und prüft vorab, ob die nötigen Voraussetzungen für einen Umzug wie richtige PHP-Versionen und Ähnliches gegeben sind.

Damit der Wechsel zu STRATO schnell gelingt, bieten wir eine **›Schritt-für-Schritt-Anleitung** an.

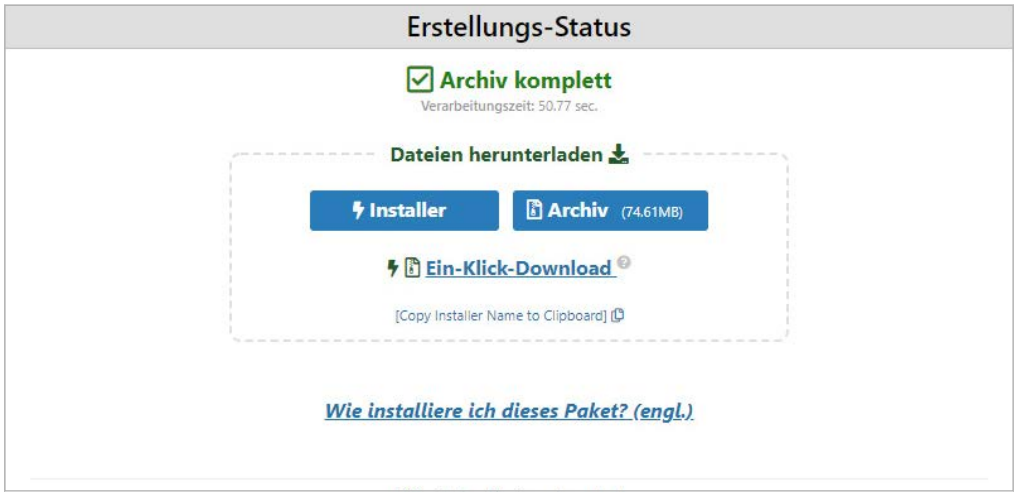
Falls Du mehrere WordPress-Instanzen installieren möchtest, solltest Du von Anfang an Unterverzeichnisse anlegen. Daneben sind für einen WordPress-Umzug aber noch einige weitere Überlegungen sinnvoll, die wir in den folgenden Tipps zusammenfassen.

1. WordPress zusammen mit der Domain umziehen

Wenn Du einen bereits vorhandenen WordPress-Blog umziehen willst, müsste die Domain theoretisch auf zwei Webservern gleichzeitig erreichbar sein. Denn Deine Domain zeigt zunächst auf den bisherigen Webspace, der Blog muss beim Umzug zum neuen Server aber möglichst schnell wieder unter dieser Domain erreichbar sein. **Wichtig**: In manchen Fällen kann es bis zu 48 Stunden dauern, bis alle DNS-Server die vorgenommenen Änderungen übernommen haben – darauf hast Du leider keinen Einfluss.

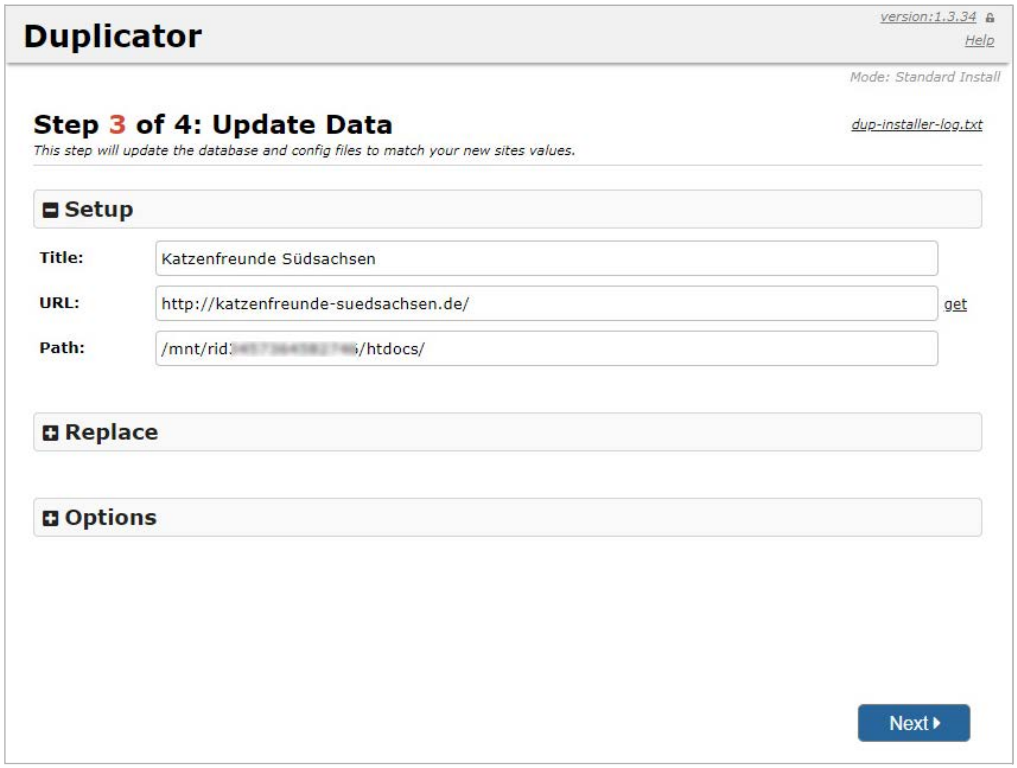
Entscheidend ist die Reihenfolge der einzelnen Schritte:

- Installiere in Deinem alten WordPress das Plugin Duplicator und lege wie in der Anleitung beschrieben ein Backup an. Lade dann das Archiv und den Installer auf Deinen Computer herunter.



- Lade das .zip-Archiv und die **installer.php** probehalber wie in der Anleitung beschrieben in den neuen Webspace. Welche Domain Du dafür verwendest, ist egal. Es handelt sich lediglich um eine Testinstallation, die Du danach wieder löschst.

- Die Installationsroutine startet, sobald Du den Installer über die neue Domain aufrufst. Den Pfad im Bereich **New Settings** musst Du manuell anpassen, da der automatisch vom Plugin ermittelte Pfad später zu Problemen führen wird. Den korrekten Webspace-Pfad (z. B. /mnt/rid/56/78/12345678/htdocs/) findest Du im STRATO Kunden-Login unter **Ihr Paket > Übersicht**. Falls Du WordPress in einem Unterverzeichnis installieren möchtest, gibst Du dieses direkt dahinter an.



- Erst wenn Du auf diese Weise getestet hast, dass alles funktioniert, ziehst Du Deine Domain vom alten Provider zu STRATO um.

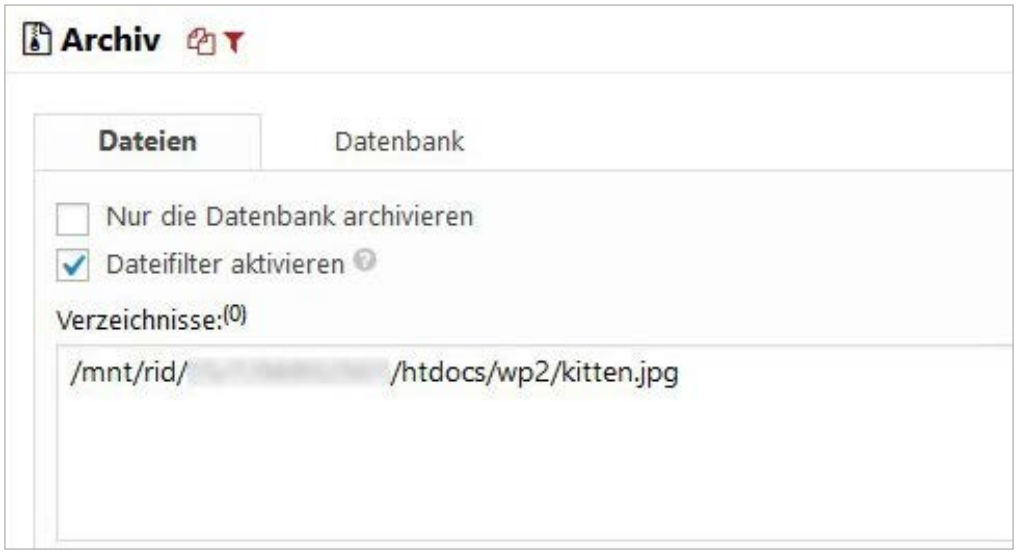
- Sobald die Domain bei STRATO erreichbar ist, spielst Du nun das Duplicator-Archiv endgültig ein und verwendest dafür Deine eigentliche Domain.

Beachte, dass Dein Blog auf dem alten Server nicht mehr erreichbar ist, sobald die Domain umgezogen ist. Je schneller Du dann das Archiv im neuen Webspace einspielst, desto kürzer ist Dein Blog offline.

2. Bilder und große Dateien manuell umziehen

Eine Stärke des Duplicator-Plugins ist der Umzug von Datenbank und Dateien in einem Rutsch. Bei kleinen Blog-Projekten funktioniert das bestens. Sobald Du aber eine größere Zahl an Bildern in WordPress hochgeladen hast oder andere große Dateien wie MP3- oder Video-Dateien vorhanden sind, empfiehlt es sich, diese separat umzuziehen.

Die Anleitung beschreibt ausführlich, wie das geht: Erstellst Du das Archiv, klammerst Du bestimmte Verzeichnisse aus und überträgst die Dateien dann per SFTP. Achte darauf, dabei aber die Ordnerstruktur nicht zu verändern, denn sonst findet WordPress die Dateien nicht mehr.

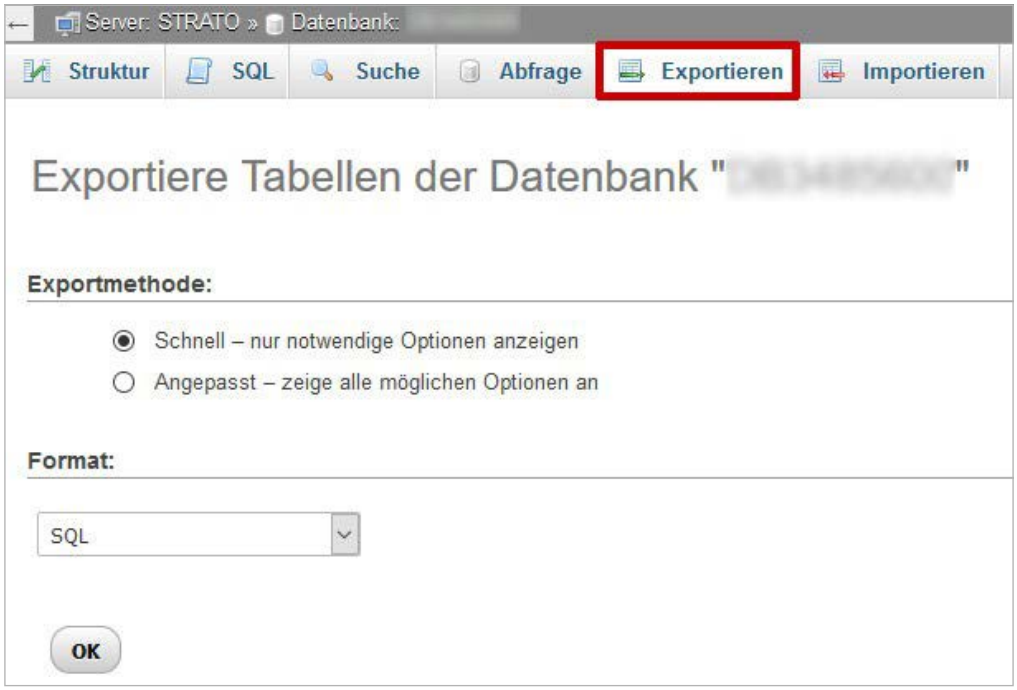


Wenn Du große Dateien auf diese Weise umziehst, statt sie in das Duplicator-Archiv aufzunehmen, erledigt das Plugin seine Aufgabe deutlich schneller und das Fehlerrisiko beim Schreiben der Archiv-Datei sinkt.

3. Auf Nummer sicher gehen

Auch wenn Du die alte WordPress-Installation vermutlich nicht mehr brauchen wirst: Lege vor dem Umzug ein Backup des gesamten Webspace (per SFTP) und der Datenbank an. Damit bist Du auf der sicheren Seite, wenn beim Umzug etwas schieflaufen sollte. Dann kannst Du zumindest den Status vor dem Umzug schnell wiederherstellen.

Tipp: Mit dem Tool **›phpMyAdmin** ist ein Datenbank-Backup besonders schnell erledigt. Die Größe der Sicherungsdatei ist auf 32 MB begrenzt. Größere Backups kannst Du per SSH anlegen.

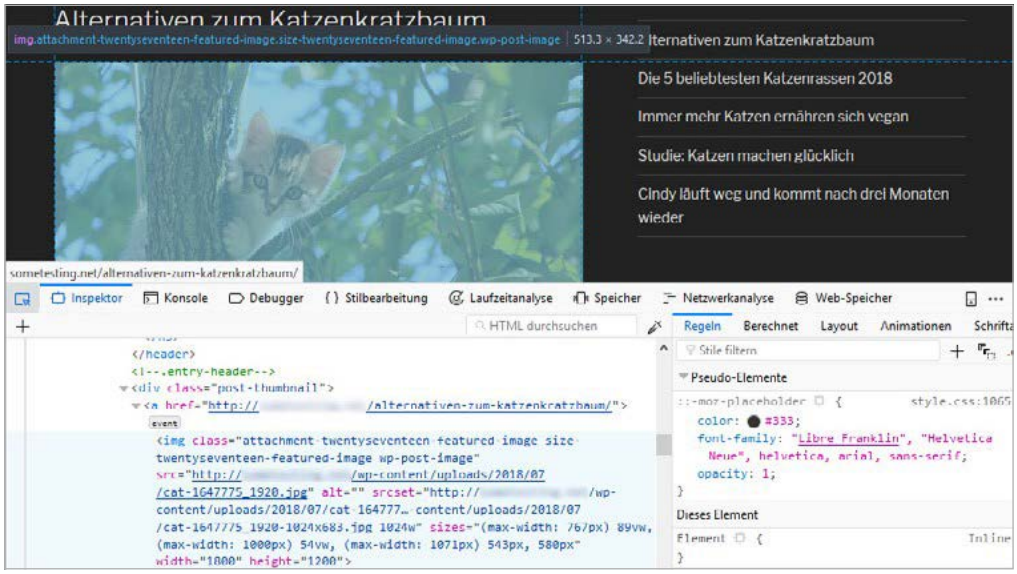


Die ursprüngliche WordPress-Installation im bisherigen Webspace solltest Du erst löschen, wenn Du absolut sicher bist, dass der Umzug funktioniert hat. Kleinere Probleme, etwa einzelne fehlende Bilder, findest Du möglicherweise erst später. Denn irgendetwas übersieht man fast immer. Ist die alte WordPress-Installation noch vorhanden, kannst Du gezielt nach Unterschieden zwischen der alten und neuen Installation suchen, um Fehlerquellen auf die Spur zu kommen.

4. Und wenn es dennoch einmal hakt …

Jeder Webserver ist ein wenig anders konfiguriert, WordPress-Themes nutzen unterschiedliche PHP-Funktionen – das kann zu Problemen beim WordPress-Umzug führen. Was kannst Du also tun, wenn nach dem Umzug irgendetwas nicht funktioniert, Bilder fehlen oder Ähnliches?

Um die möglichen Fehler einzugrenzen, solltest Du Dir zunächst den HTML-Quellcode im Browser ansehen und prüfen, ob dort beispielsweise falsche Verzeichnispfade vorkommen. Webbrowser haben meist eine eingebaute Entwickler-Funktion, über die Du den Quellcode analysieren kannst. In Firefox und Chrome zum Beispiel rufst Du diese über die Taste F12 auf.



Entwicklerwerkzeuge in Firefox

Hast Du beispielsweise einen falschen Verzeichnispfad gefunden, kannst Du das über ein Suchen-und-Ersetzen-Plugin korrigieren. Liegt der Ursprung des Problems in einer der Theme-Dateien, lädst Du das komplette Theme-Verzeichnis per SFTP auf Deinen Computer und benutzt die dateiübergreifende Suchfunktion Deines Editors (z. B. **›Notepad++**). Einmal gefunden, kannst Du den Fehler direkt in der Theme-Datei korrigieren.

5. Dem Google-Ranking zuliebe nicht zu viel gleichzeitig ändern

Die Versuchung ist groß, bei einem Server-Umzug auch all die Dinge zu verändern, die man eh schon lange einmal anpacken wollte. Doch genau das solltest Du vermeiden. Behalte auf jeden Fall erst einmal Deine Domain bei, übernimm die bisherige URL-Struktur und verändere das Theme nicht. Das hat zwei handfeste Vorteile:

1. Je weniger Du veränderst, desto weniger potenzielle Fehlerquellen gibt es. Gerade bei mehreren, vermeintlich voneinander unabhängigen Veränderungen entstehen oft Wechselwirkungen, die nur schwer zu identifizieren sind.

2. Suchmaschinen reagieren empfindlich auf globale Änderungen von Websites. Durch den Umzug verändert sich ohnehin schon die IP-Adresse Deines Blogs. Das ist für Google ein Signal, für eine Weile genauer hinzusehen, ob die Domain womöglich verkauft wurde und nun vielleicht als Spam-Schleuder dient. Würdest

Du jetzt auch noch das Design – und damit die HTML-Struktur – verändern, wäre das für Google ein weiteres Alarmsignal. Wählst Du eine neue Permalink-Struktur, verändern sich die URLs, was ebenfalls negative SEO-Folgen haben kann.

War Dein Umzug erfolgreich, kannst Du nach ein paar Wochen die nächsten Veränderungen in Angriff nehmen. Aber auch hier gilt: Lieber in kleinen Schritten, um keine Probleme mit Suchmaschinen zu riskieren.

6. Geduld und gute Vorbereitung vermeiden Ärger

Ein Tipp zum Schluss: Auch wenn der Umzug von WordPress verhältnismäßig einfach ist und in den meisten Fällen glattläuft, bleibt doch immer ein Restrisiko. Um das so gering wie möglich zu halten, solltest Du sehr konzentriert arbeiten und geduldig sein. Wirft die Prüfung des Duplicator-Plugins eine Warnung aus oder sind Dir Details beim Ablauf unklar, recherchiere und teste vorab so lange, bis alle Zweifel ausgeräumt sind. Die eigene Ungeduld verleitet oft zur Unachtsamkeit. Bedenke daher: Geduld spart gerade bei einer Aktion wie dem Umzug eines WordPress-Blogs eine Menge Ärger.

1.3 Lokale Testumgebung erstellen

Neue Funktionen, Plugins oder Designs kannst Du entweder direkt online oder in einer lokalen Testumgebung testen. Letzteres ist sicherer, weil Du unabhängig von der Website arbeitest und Dir keine Gedanken über Sicherheitsrisiken machen musst. Wir zeigen Dir im Folgenden, wie Du eine einfache lokale Testumgebung für WordPress erstellst.

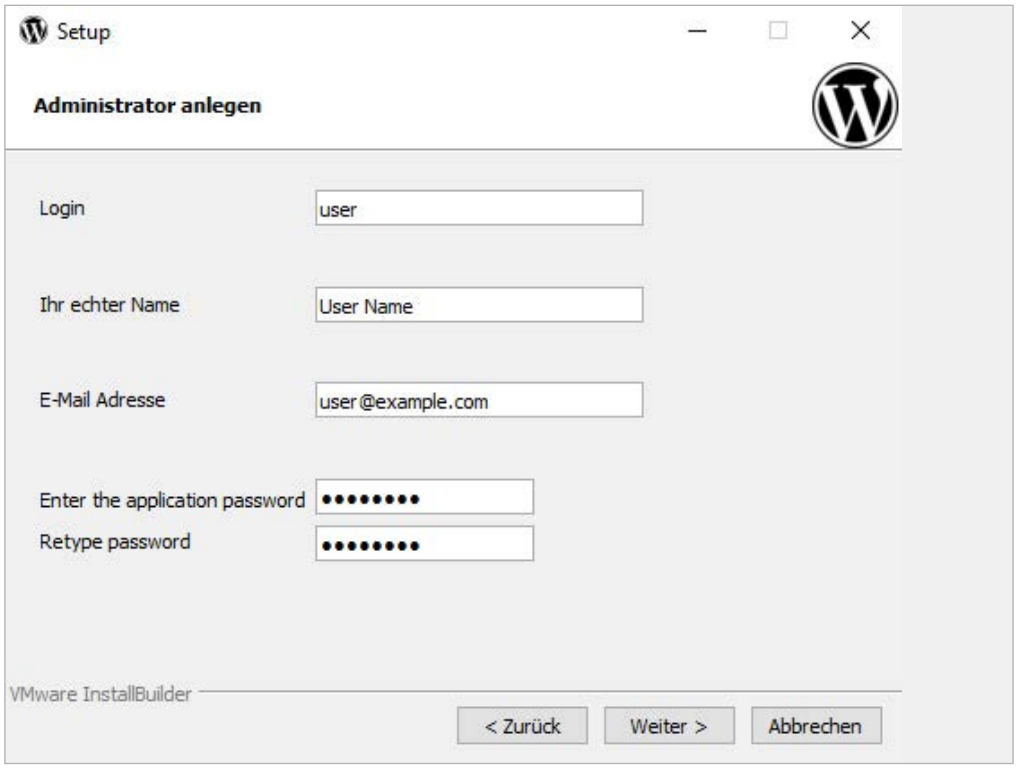
Wer den **›White Screen of Death** kennt, kann ein Lied davon singen: Themes und Plugins live auszuprobieren, ist keine gute Idee. Selbst gut gepflegte Software kann im Zusammenspiel mit anderen Programmen Probleme verursachen. Riskant sind vor allem Codeanpassungen im laufenden Betrieb. Und wer ein eigenes Theme entwickeln möchte, macht dies ohnehin außerhalb der Produktivumgebung.

Folglich solltest Du Anpassungen immer vorab (offline) testen. Für die Entwicklung eines eigenen Themes sind das Starter-Theme Underscores und das Framework Bootstrap 4 eine ideale Basis.

Schritt 1: XAMPP und WordPress einrichten

Zunächst installierst Du WordPress lokal auf Deinem Computer. Um Server und Datenbank zu emulieren verwendest Du am besten **›Xampp mit einer aktuellen PHP Version (ab 7.4)**. Um potenzielle Probleme mit fehlenden Berechtigungen zu umgehen, solltest Du das Programm nicht im vorgeschlagenen Betriebssystem-Verzeichnis installieren (sondern z. B. unter »Dokumente«). Nach der Installation startest Du die beiden Module Apache und MySQL.

Anschließend kannst Du eine Datenbank anlegen und WordPress manuell installieren. Schneller geht es, wenn Du das WordPress Modul verwendest. **›Lade dazu die Datei herunter** und starte die Installation per Doppelklick. Nachdem Du den XAMPP-Ordner angegeben hast, kann Du einen WordPress-Benutzer anlegen und danach den Blog-Namen eingeben:



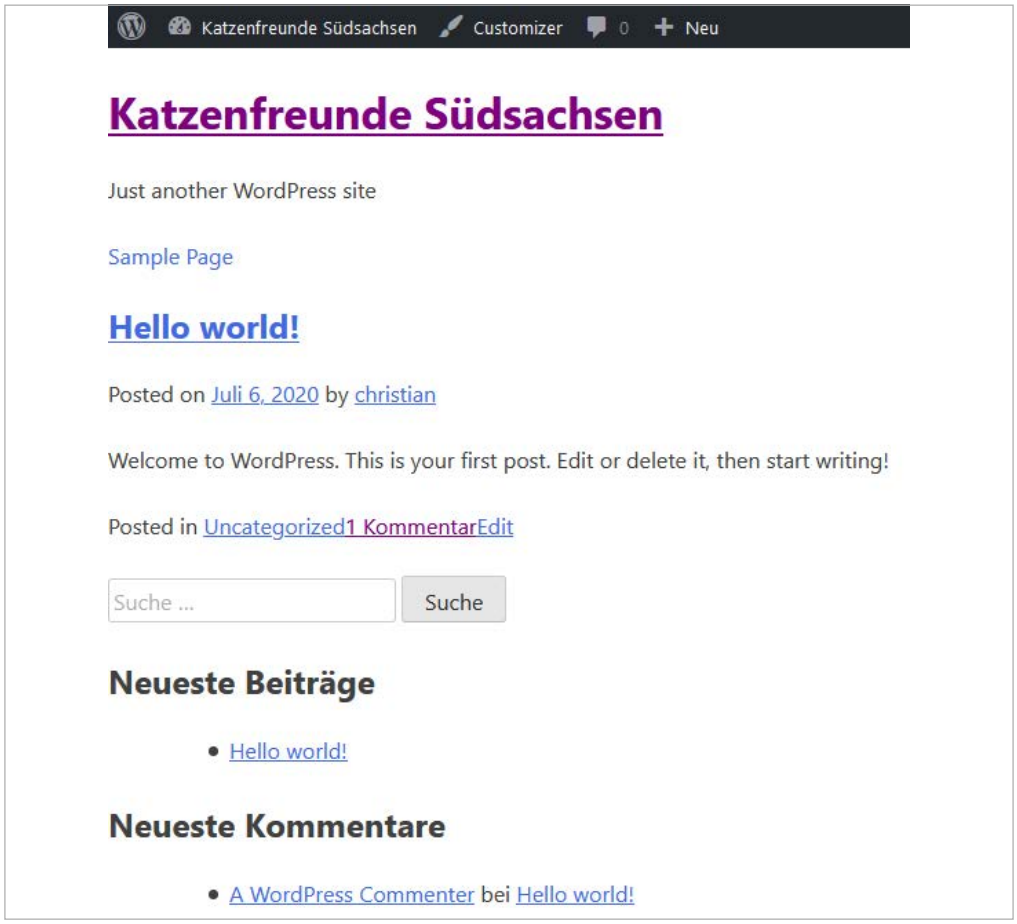
Logge Dich anschließend über **›http://127.0.0.1/wordpress/wp-admin/** in das WordPress-Backend ein und führe erstmal alle angezeigten Aktualisierungen durch.

Schritt 2: Theme installieren

Um Anpassungen an Deiner bestehenden Website vorab zu testen, installierst Du auf Deiner Testumgebung das gleiche Theme und übernimmst dieselben Einstellungen Deiner Website. Ist die Testumgebung für die Entwicklung eines eigenen Themes gedacht, nutzt Du am besten das Basis-Theme **›Underscores**. Damit hast Du schon mal ein sauber programmiertes Fundament für Dein eigenes Layout. Gib dem Theme einen Namen (z. B. »meintheme«), klicke auf **Generate** und lade dann die .zip-Datei herunter.

Basis-Theme per Knopfdruck: Underscores

Nachdem Du das Theme installiert und aktiviert hast, kannst Du Dir das Basis-Layout über **› http://127.0.0.1/wordpress/wp-admin/** anschauen:



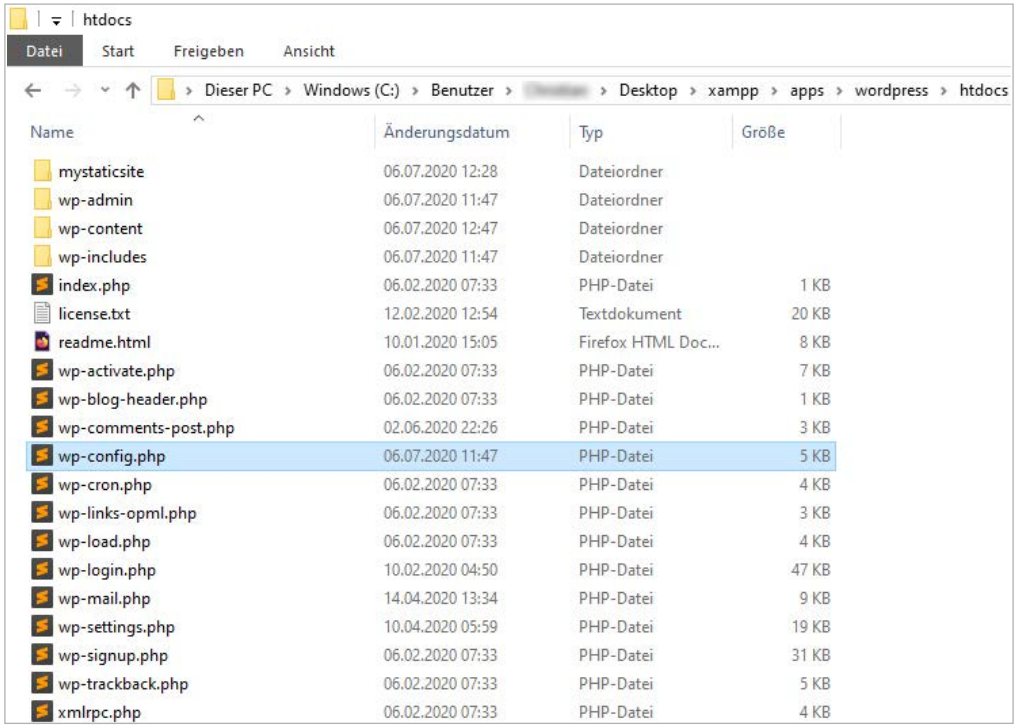
Das Layout von Underscores ist im Originalzustand bewusst minimalistisch. Damit bleibt mehr Raum für individuelle Anpassungen.

Aus Sicherheitsgründen sollte mindestens ein weiteres Theme installiert sein, auf welches WordPress im Notfall zurückgreifen kann.

Schritt 3: Debug-Modus aktivieren

Zu einer Testumgebung gehört auch das sogenannte Debugging, also das Aufspüren und Beheben von Bugs. Damit WordPress bei fehlerhaftem PHP-Code statt einer weißen Seite eine aussagekräftige Fehlermeldung ausgibt, musst Du den Debug-Modus aktivieren.

Navigiere dazu in das WordPress-Hauptverzeichnis innerhalb des XAMPP-Ordners. Wurde XAMPP z. B. unter Windows auf dem Desktop installiert, kann der Pfad so aussehen: **C:\Users\NUTZERNAME\Desktop\xampp\apps\wordpress\htdocs**



Hier siehst Du das WordPress-Hauptverzeichnis innerhalb des XAMPP-Ordners unter Windows 10.

Ersetze in der Datei **wp-config.php** die Zeile

```
define(‚WP_DEBUG‘, false);
```

durch diese:

```
define(‚WP_DEBUG‘, true);
```

Wichtig: Denke daran, diese Einstellung wieder auf **false** zurückzusetzen, bevor Du das fertige Theme in Deinem Blog online nimmst.

Schritt 4: Bootstrap einbinden

Bootstrap ist ein beliebtes Open Source Framework mit fertigen Buttons, Navigationsleisten, Grid-Systemen, Tabellen und vielem mehr. Als Basis dienen

CSS-Komponenten, Schriften, Icons und JavaScript-Code. Die benötigten Dateien und Code-Blöcke können frei verwendet werden. Um den Baukasten nutzen zu können, musst Du Bootstrap mit WordPress verbinden.

Dazu lädst Du **›Bootstrap in der aktuellen Version** herunter (Compiled CSS and JS). Entpacke die Datei und benenne den ersten Unterordner (in unserem Beispiel „bootstrap-4.5.0-dist“) in »bootstrap« um. Lade diesen Ordner anschließend in das Theme-Verzeichnis (xampp\apps\wordpress\htdocs\wp-content\themes).

Um WordPress und Bootstrap miteinander zu verknüpfen, musst Du jetzt nur noch die **functions.php** Deines Themes anpassen.

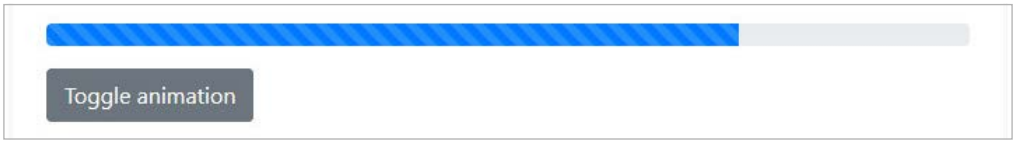
Hinweis: Die nachfolgenden Zeilenangaben für den Code können von denen Deiner Underscores-Version abweichen.

Um Zeile 145 befindet sich **function meintheme_scripts()** – statt »meintheme« steht dort der Name Deines Themes. Darunter folgen mehrere Funktionen. Füge vor der schließenden geschweiften Klammer } die folgenden Zeilen ein. Achte dabei darauf, im Code die jeweils gültige Bootstrap-Version anzugeben (in diesem Beispiel 4.5.0):

```
// Bootstrap
    wp_register_script( ‚bootstrap-js‘, get_template_directory_uri() .
‚/bootstrap/js/bootstrap.min.js‘, array( ‚jquery‘ ), ‚4.5.0‘, true );
    wp_register_style( ‚bootstrap-css‘, get_template_directory_uri() .
‚/bootstrap/css/bootstrap.min.css‘, array(), ‚4.5.0‘, ‚all‘ );
    wp_enqueue_script( ‚bootstrap-js‘ );
    wp_enqueue_style( ‚bootstrap-css‘ );
```

Im Editor sieht das so aus:

Nach dem Upload ist das Theme mit Bootstrap verknüpft. Damit kannst Du Bootstrap-Elemente in Deinem Blog testen: Kopiere einfach die vorgefertigten Code-Blöcke von **›getbootstrap.com** und füge sie in den Code-Editor ein: **Beiträge > Erstellen > Code-Editor** (Drei-Punkte-Menü oben rechts). Nach dem Veröffentlichen erscheinen die Elemente im Frontend Deines Blogs.



Zu den Bootstrap-Komponenten gehören zum Beispiel animierte Fortschrittsbalken.

Spielwiese für WordPress

Ob Theme-Entwicklung oder -Anpassungen: Mit den vier gezeigten Schritten kannst Du sämtliche Änderungen Deiner privaten Website oder Deines Blogs bequem und gefahrlos vorab testen. Achte darauf, dass Du identische Einstellungen wählst und WordPress, Theme und Plugins auf dem gleichen aktuellen Stand sind. Wenn alles reibungslos funktioniert, lädst Du die Dateien hoch. Im nächsten Kapitel erfährst Du, wie Du das Layout flexibel weiterentwickelst.

2 Layout

2.1 Child-Themes: Individuelles WordPress-Design ganz einfach

Fertige Website-Vorlagen – Themes – gibt es massenhaft. Doch selten findet sich eines, das exakt passt. Fast immer sind kleine Anpassungen nötig – und seien es nur die Schriftart, ein paar Farben oder eine zu dick geratene Linie. Schon mit HTML- und CSS-Grundkenntnissen kannst Du WordPress-Themes ganz einfach verändern und anpassen. Für größere Änderungen ist Basis-Wissen in PHP hilfreich.

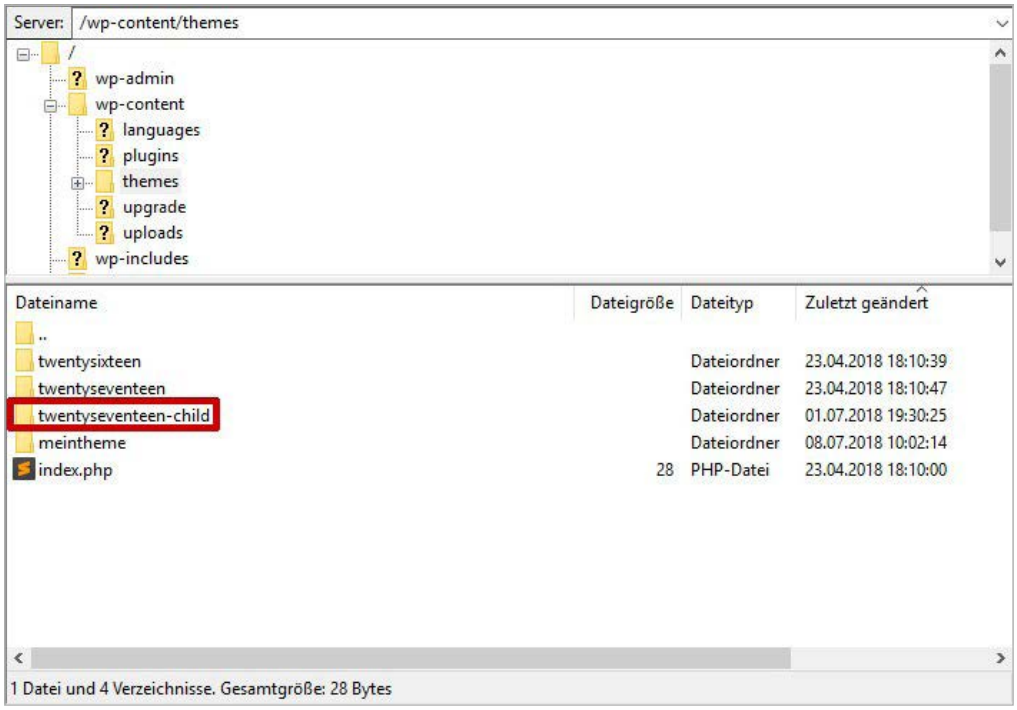
Mit sogenannten »Child-Themes« bietet WordPress ein mächtiges Feature. Sie setzen auf ein vorhandenes Theme mit dessen Formatierungen und Funktionen und ergänzen oder modifizieren es mit eigenen Elementen und Designs. Der große Vorteil: Das Original-Theme bleibt dabei unangetastet, Updates des Originals beeinträchtigen die Anpassungen im Child-Theme nicht.

Schritt für Schritt: Erstelle Dein eigenes Child-Theme

Ein wenig Aufwand bringt nur das erstmalige Anlegen des Child-Themes mit sich. Danach sind die Anpassungen genau so unkompliziert, als würdest Du direkt im Original-Theme arbeiten. Zunächst suchst Du Dir ein Theme aus, das Deinen Vorstellungen so nahe wie möglich kommt, und installierst es. Achte darauf, dass es als geeignet für Child-Themes beschrieben wird. Die meisten Themes sind das, aber es gibt Ausnahmen. Im Folgenden beschreiben wir, wie Du ein Child-Theme manuell anlegst.

1. Ordner fürs Child-Theme anlegen

Themes liegen in Deinem Webspace im Verzeichnis **wp-content/themes**.
Wir verwenden für unser Beispiel hier ein Parent-Theme, das im Verzeichnis **wp-content/themes/parenttheme** liegt. Entsprechend musst Du bei den folgenden Beschreibungen **parenttheme** jeweils gegen den Verzeichnisnamen Deines tatsächlichen Parent-Themes austauschen.



Der Child-Theme-Ordner liegt auf derselben Verzeichnis-Ebene wie das Parent-Theme.

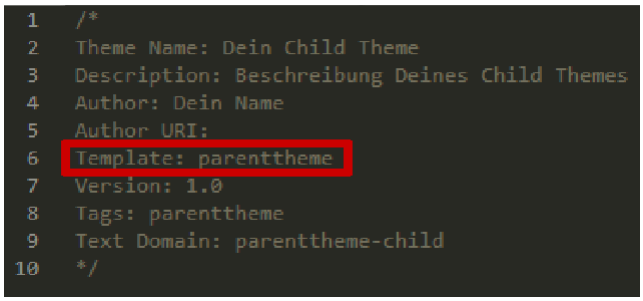
Unter **wp-content/themes** legst Du nun also auf der gleichen Ebene wie das ausgewählte Parent-Theme ein neues Verzeichnis an. Wie Du das Verzeichnis benennst, ist egal. Es bietet sich aber zum späteren Verständnis beipielsweise **parenttheme-child** als Name an.

2. Child-Theme definieren

In diesem neuen Verzeichnis – im Beispiel **wp-content/themes/parenttheme-child** – legst Du eine neue Datei namens style.css an. Sie muss die folgenden Code-Zeilen beinhalten, damit WordPress erkennt, dass es sich hier um ein Child-Theme zum Theme im Verzeichnis **wp-content/themes/parenttheme** handelt:

```
/*
Theme Name: Dein Child-Theme
Description: Beschreibung Deines Child-Themes
Author: Dein Name
Author URI:
Template: parenttheme
Version: 1.0
Tags: parenttheme
Text Domain: parenttheme-child
*/
```

Entscheidend ist die Zeile **Template**, denn dort muss der exakte Name des Verzeichnisses eingetragen sein, in dem das ausgewählte Parent-Theme liegt. In unserem Beispiel also **parenttheme**.



Das Child-Theme funktioniert nur, wenn das Parent-Template korrekt benannt ist.

3. Stylesheets des Parent-Themes verknüpfen

Lege nun im Hauptverzeichnis Deines Child-Themes eine neue Datei namens **functions.php** an. Diese Datei brauchst Du, um die CSS-Definitionen aus dem Parent-Theme mit dem Child-Theme korrekt zu verknüpfen und später gegebenenfalls auch für PHP-Anpassungen des Themes. Füge folgende Zeilen in die **functions.php** ein. Damit legst Du fest, dass zunächst das Stylesheet des Parent-Themes und dann das des Child-Themes geladen wird:

```
<?php
add_action( ‚wp_enqueue_scripts‘, ‚theme_enqueue_styles‘ );
function theme_enqueue_styles() {
   wp_enqueue_style( ‚parent-style‘, get_template_directory_uri() . ‚/style.css‘ );
   wp_enqueue_style( ‚child-style‘, get_stylesheet_directory_uri() . ‚/style.css‘,
array(‚parent-style‘) );
}
```

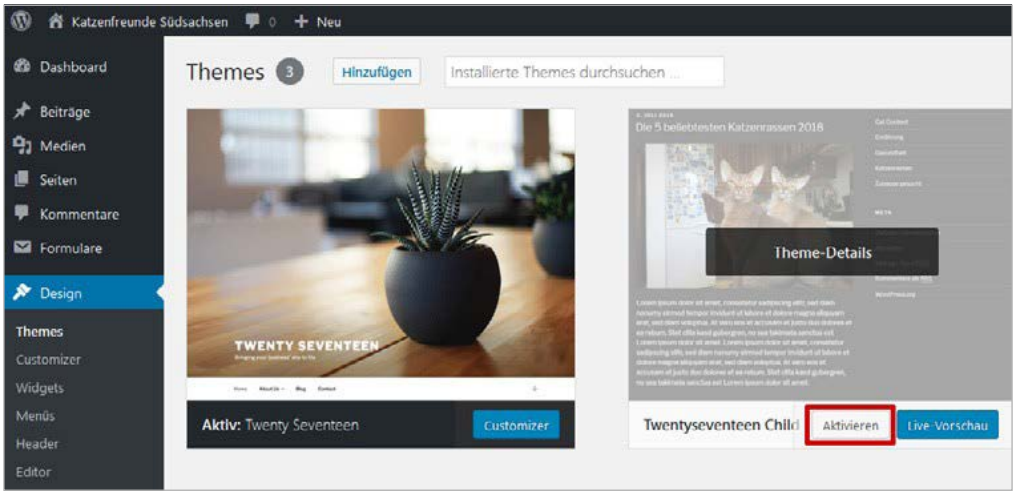
Übrigens: Die früher verwendete Methode, die CSS-Datei des Parent-Themes über @import in der style.css zu integrieren, wird von WordPress nicht mehr empfohlen. Sie hat vor allem Performance-Nachteile.

Falls diese Methode bei Deinem Theme nicht funktioniert, findest Du **›hier eine alternative Anleitung**.

4. Child-Theme aktivieren

Bevor Du das das Theme aktivierst, kannst Du noch ein Vorschaubild für die Themes-Übersicht im Backend einfügen. Erstelle dazu einen Screenshot Deines Child-Themes und lege diesen in der Größe 880×660 Pixel mit dem Dateinamen **screenshot.jpg** ins Hauptverzeichnis Deines Child-Themes. WordPress erkennt das Vorschaubild automatisch und stellt Dir nun auch Dein Child-Theme in der Übersicht optisch ansprechend dar. Diesen Schritt kannst Du natürlich erst dann gehen, wenn Du alle Anpassungen durchgeführt hast.

Jetzt ist das Child-Theme grundsätzlich einsatzfähig, sodass Du es nun hochladen und in der Admin-Oberfläche im Menüpunkt **Design > Themes aktivieren** kannst.



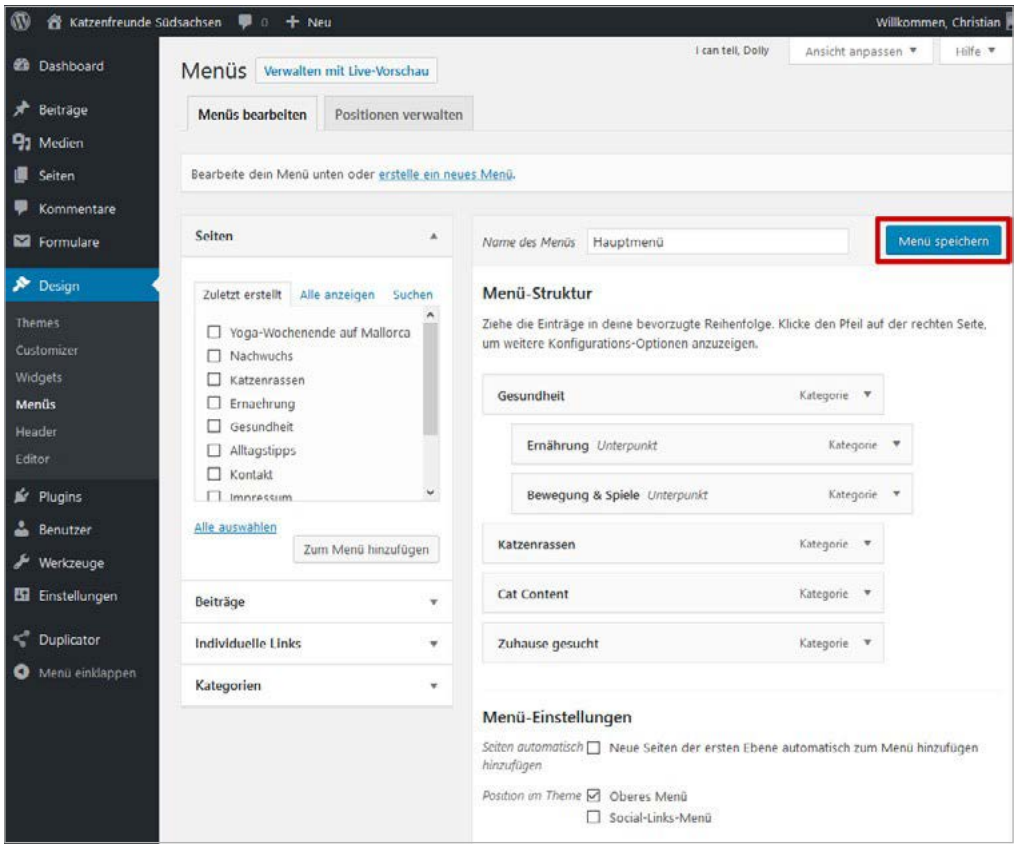
Ein Klick genügt und Dein neues Child-Theme ist aktiv.

Erst einmal ändert sich damit gar nichts, jedenfalls nichts Sichtbares. Denn noch hast Du ja keine Änderungen im Child-Theme vorgenommen. Um zu sehen, ob Du erfolgreich warst, öffne die **style.css** und füge dort eine neue Formatierung ein, beispielsweise:

```
body {color: blue !important;
}
```

Nachdem Du die veränderte **style.css** auf dem Server gespeichert hast, sollte nun der Fließtext auf Deiner Website in Blau erscheinen.

Falls Du in WordPress mit eigenen Menüs arbeitest, gibt es nach der Aktivierung eines Child-Themes manchmal Probleme. Die behebst Du ganz einfach, indem Du unter **Design > Menüs** das jeweilige Menü auswählst und einmal auf **Menü speichern** klickst.



Sofern vorhanden, solltest Du individuelle Menüs nach Aktivierung des Child-Themes einmal neu speichern.

Systematik von Child-Themes

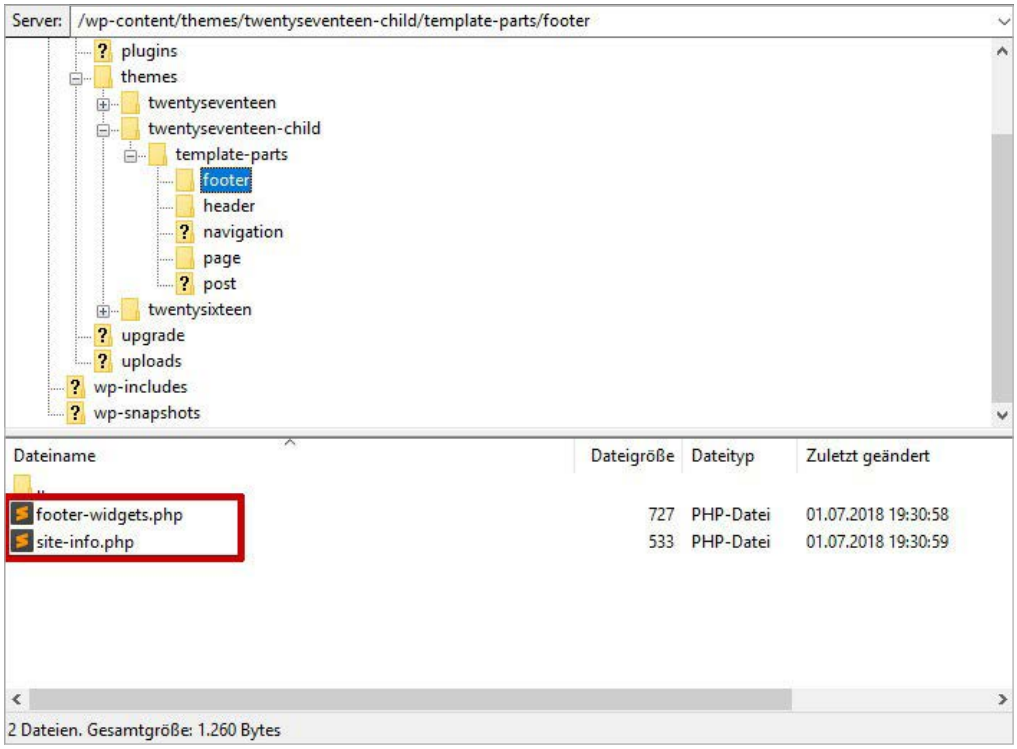
Die beiden Dateien **style.css** und **functions.php** haben in Child-Themes eine Sonderstellung: Die Einträge in der **style.css** lädt WordPress zusätzlich zu den äquivalenten Dateien des Parent-Themes. Enthält das Child-Theme also Formatierungen für dieselben Elemente wie das Parent-Theme, werden letztere durch das Child-Theme überschrieben.

Bei der **functions.php** lädt dagegen immer zuerst die Datei des Child-Themes, danach die des Parent-Themes. Funktionen mit gleichem Namen werden vom Parent-Theme also überschrieben. Du solltest daher darauf achten, dass die Funktionen in Deinem Child-Theme keine Namen benutzen, die bereits in der **functions.php** des Parent-Themes vorkommen. Ein Hinweis für erfahrene Anwender: Gleichnamige Funktionen können im Child-Theme die ursprünglichen Funktionen komplett ersetzen, indem Du sie über **remove_action()** beziehungsweise **remove_filter()** von dem Hook löst, dem sie zugeordnet sind, oder indem Du im Child-Theme eine höhere Priorität festlegst.

Alle anderen Dateien, die Du in Deinem Child-Theme einsetzt, ersetzen die äquivalenten Dateien des Parent-Themes komplett. Sie werden anstatt und nicht zusätzlich zu den Dateien aus dem Parent-Theme geladen. WordPress prüft also, ob eine entsprechende Datei im Child-Theme vorhanden ist und gibt selbiger gegebenenfalls den Vorzug.

> Liegen die Original-Dateien in einem Unterverzeichnis, müssen die entsprechenden Dateien im Child-Theme ebenfalls in einem identischen Unterverzeichnis liegen.

Beispiel: Die Dateien **footer-widgets.php** und **site-info.php** des Themes Twenty Seventeen liegen im Unterverzeichnis **/wp-content/themes/twentyseventeen/template-parts/footer**. Um sie zu ersetzen, müssen die entsprechenden Dateien des Child-Themes einen äquivalenten Pfad haben: **/wp-content/themes/twentyseventeen-child/template-parts/footer**.



Alle Dateien mit Ausnahme von style.css und functions.php ersetzen die Dateien des Parent-Themes komplett.

Theme-Updates prüfen

Du hast jetzt alle Möglichkeiten in der Hand, um Gestaltung und Funktionalität Deines WordPress-Themes zu verändern, ohne das zu Grunde liegende Parent-Theme anzufassen. Dieses kannst Du also bedenkenlos aktualisieren, wenn Updates dazu erscheinen.

Aber natürlich entwickelt sich jedes Theme weiter, sodass Updates auch Veränderungen betreffen können, die Du im Child-Theme gemacht hast. Dateien aus dem Parent-Theme, die Du übernommen und verändert hast, könnten Sicherheitslücken enthalten, die auch beim Update des Parent-Themes nicht behoben werden, weil WordPress weiterhin die – dann veraltete – Kopie in Deinem Child-Theme verwendet.

Deshalb ist es wichtig, dass Du bei jedem Update des Parent-Themes in dessen Dokumentation schaust und prüfst, was sich dort verändert hat, um notfalls auch Dein Child-Theme anzupassen.

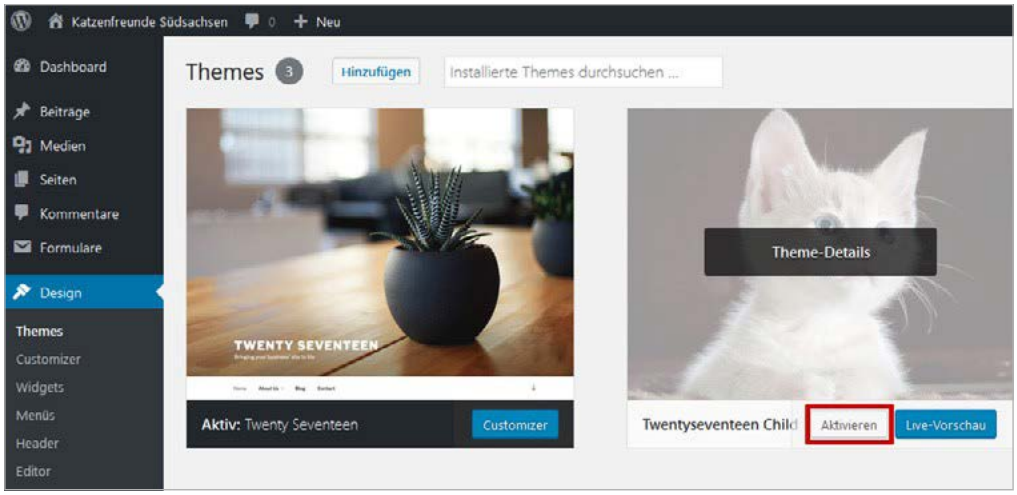
Minimal-Prinzip

Der wichtigste Tipp für Child-Themes lautet daher: Beschränke Dich bei Veränderungen auf das absolut Nötige. Je weniger Parent-Theme-Dateien Du übernimmst und je weniger Veränderungen Du am Design vornimmst, desto unwahrscheinlicher sind spätere Konflikte – und desto unwahrscheinlicher ist es, dass Du bei Theme-Updates Anpassungen vornehmen musst.

Das geringste Konfliktpotenzial haben die CSS-Anweisungen in der **style.css**. Wenn Du also damit auskommst, ausschließlich hier Anpassungen vorzunehmen, kannst Du Dir eine umfangreichere Prüfung von Theme-Updates sparen.

Viel Spaß beim Gestalten!

Child-Themes werden gerne unterschätzt, sind aber ein sehr mächtiges und praktisches Feature von WordPress. Du hast damit alle Freiheiten, das Theme anzupassen, ohne das Haupt-Theme zu verändern. Beachten musst Du lediglich, dass zu viele und zu umfassende Veränderungen irgendwann dennoch zu Konflikten führen werden. In solchen Fällen kann es sinnvoll sein, lieber gleich zu einem anderen Theme zu wechseln, das schon in seiner Grundstruktur besser zu Deinen Anforderungen passt. Für kleinere Design-Anpassungen sind Child-Themes dagegen sehr unkompliziert und sicher.



2.2 5 Schönheitstipps für den Kommentarbereich

Den Kommentarbereich der meisten WordPress-Themes kann man als nüchtern-funktional beschreiben. Das ist praktisch, aber wenig einladend. Doch was macht einen ansprechenden Kommentarbereich aus? Zum Beispiel eine eigene Überschrift für das Formular, hervorgehobene Autoren-Kommentare, individuelle Farben, eine Paginierung für Beiträge mit vielen Kommentaren und die Möglichkeit, Kommentare zu liken. Wenn Du das alles willst, solltest Du weiterlesen.

So sehen Kommentare bei Twenty Twenty standardmäßig aus.

Tipp 1: Kommentare hervorheben

Um die Kommentare und die Formularfelder farblich vom Hintergrund abzuheben, passt Du diese einfach per CSS an. Sämtliche Anweisungen kommen wie gewohnt in die **style.css** des Child-Themes.

```
/* Leser-Kommentare hervorheben */
.comment-body {
    background: #f8f8f8;
}
```

Praktischerweise vergibt WordPress für Autoren-Kommentare eine eigene Klasse namens **.bypostauthor**. Damit kannst Du diese etwa durch einen seitlichen Rahmen von den anderen abgrenzen:

```
/* Autoren-Kommentare hervorheben */
.bypostauthor > article {
   border-right: #cd2653 solid 10px;
}
```

Der Kommentar des Autors Christian wird durch einen einseitigen Rahmen auf der rechten Seite hervorgehoben.

Tipp 2: Überschrift für den Kommentarbereich ändern

Die Überschrift lautet standardmäßig »Schreibe einen Kommentar«. Um das zu ändern, lädst Du im Theme-Verzeichnis die Datei **comments.php** herunter und fügst in das Array der Funktion comment_form(); folgende Zeilen ein:

```
// „Hinterlasse einen Kommentar“ ändern
   ‚title_reply‘ => ‚Sag mir Deine Meinung!‘,
```

Im Editor sieht das so aus:

```
comment_form(
    array(
        'class_form'         => 'section-inner thin max-percentage',
        'title_reply_before' => '<h2 id="reply-title" class="comment-reply-title">',
        'title_reply_after'  => '</h2>',
        // Überschrift für den Kommentarbereich ändern
        'title_reply' => 'Sag mir Deine Meinung!',
    )
);
```

Statt »Sag mir Deine Meinung!« kannst Du natürlich auch irgendetwas anderes einfügen. Anschließend lädst Du die Datei in das Hauptverzeichnis Deines Child-Themes.

Tipp 3: Feld für Website entfernen

Mit den folgenden Zeilen entfernst Du das Formularfeld für die Website aus dem Kommentarbereich. Füge den Code einfach in die **functions.php** Deines Child-Themes ein:

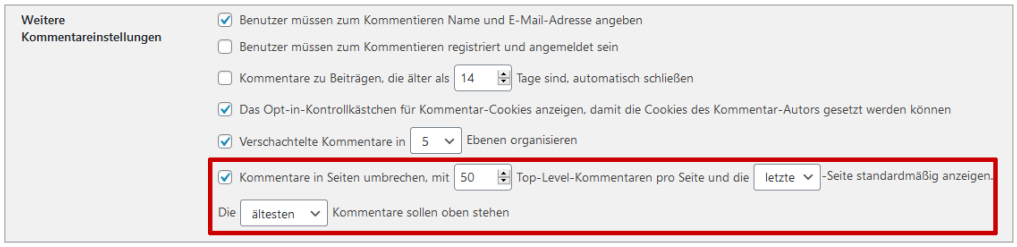
```
//Website-Feld aus Kommentarbereich entfernen
add_filter(‚comment_form_default_fields‘, ‚remove_website_field‘);
function remove_website_field($fields){
   if(isset($fields[‚url‘]))
   unset($fields[‚url‘]);
   return $fields;
}
```

Nun wird der Cookie-Hinweis mit der Checkbox nicht mehr korrekt dargestellt. Um das zu fixen, schreibst Du noch folgende Anweisung in die style.css:

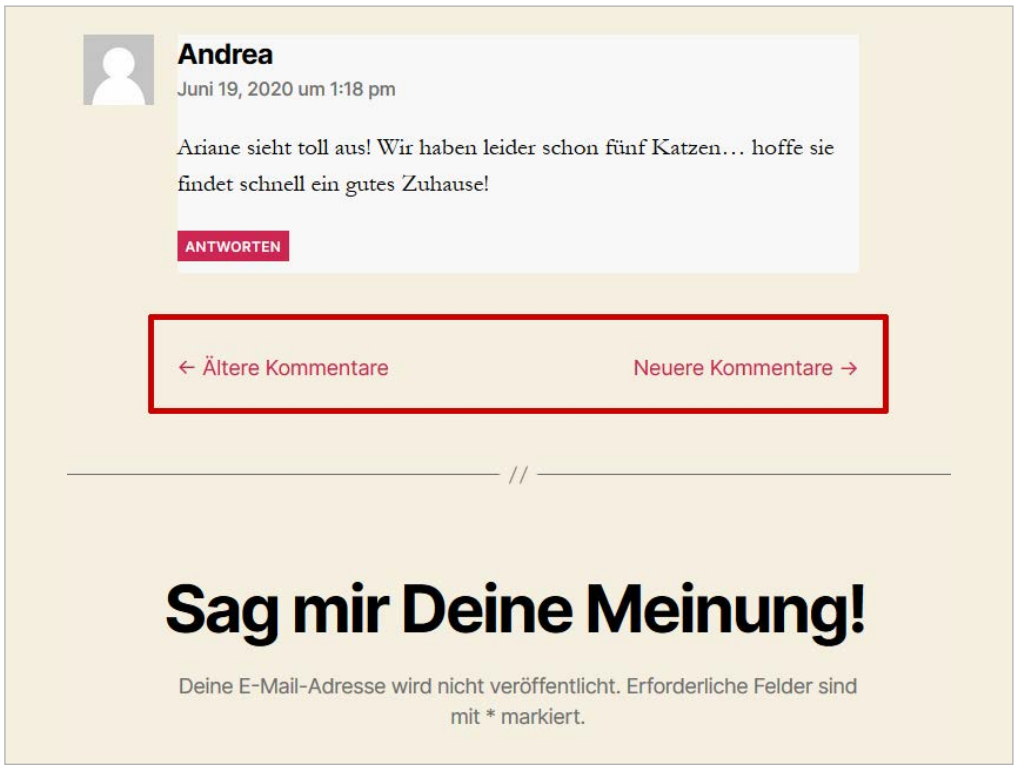
```
* Cookie-Hinweis ausrichten */
.comment-form-cookies-consent {
   display: inline-flex;
}
```

Tipp 4: Formular bunter machen

Unter einer hohen Zahl an Kommentaren leiden Performance und Übersicht. In dem Fall ist eine Aufteilung des Kommentarbereichs auf mehreren Seiten sinnvoll. Dazu setzt Du im Backend unter **Einstellungen > Diskussion** ein Häkchen bei »Kommentare in Seiten umbrechen (...)«.

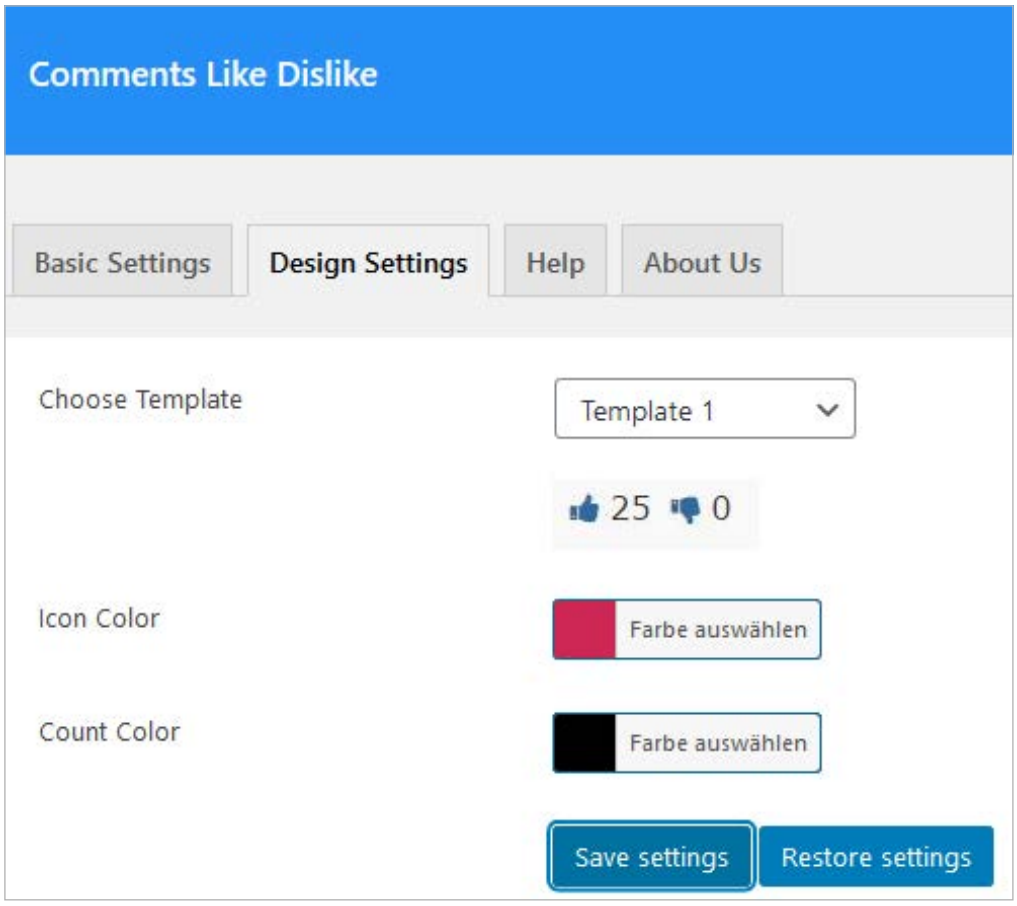


Im Frontend erscheinen nun unterhalb des Kommentarbereichs Links zu älteren und neueren Kommentaren. Damit können Deine Besucher das Feedback in Deinem Blog besser nachvollziehen.

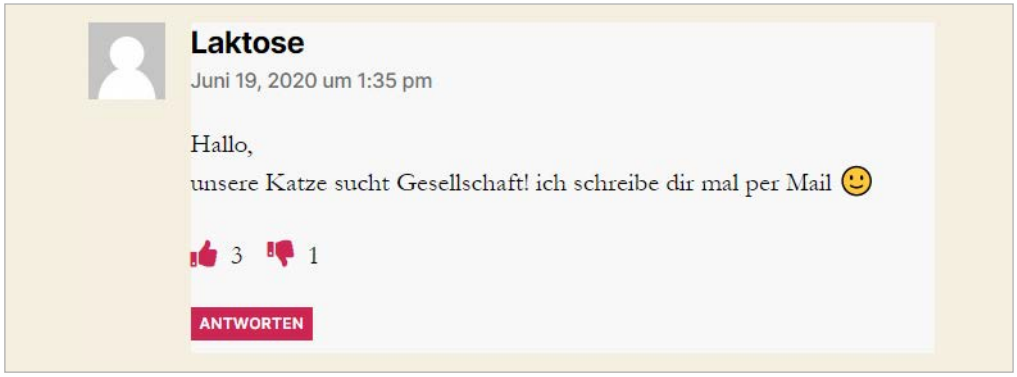


Tipp 5: Like-Buttons hinzufügen

Liken und Favorisieren gehören zum digitalen Alltag. Dennoch: Obwohl das Bewerten anderer Posts die Interaktion steigert, verzichten die meisten Blogs auf diese Funktion. Dabei ist es mit dem **›Plugin Comments Like Dislike** so einfach. Nachdem Du das Plugin installiert und aktiviert hast, kannst Du in den Plugin Settings unter dem Reiter **Design Settings** sogar Templates und die Farbe auswählen.



Wir haben uns in diesem Beispiel für Daumen in der Standard-Farbe des Themes Twenty Twenty (#cd2653) entschieden:



2.3 Autor-Template erstellen

Ein Autor-Template bestimmt, wie die Archiv-Seite Deiner Beiträge aussieht. Wenn Du in einem Beitrag auf Deinen Namen klickst, findest Du in der Regel lediglich eine chronologische Auflistung der letzten Beiträge. In nur wenigen Schritten verwandelst Du Dein Archiv in eine individuelle Autorenseite mit Bild, biografischen Infos und Kontaktdaten.

1. Template erstellen

Erstelle dazu eine neue Datei namens **author.php**. Was Du dort eingibst, hängt vom verwendeten Theme und Deinen Vorstellungen ab. In unserem Beispiel soll die (bei Twenty Twenty) unterhalb der Beiträge angezeigte Autoren-Bio auch im Autoren-Template sichtbar werden. Darunter fügen wir das Feld für die Webseite und die E-Mail-Adresse ein. Der komplette Code des Templates sieht dann so aus:

```
<?php
/**
   * Autor Page Template
   *
   * @package WordPress
   * @subpackage Twenty_Twenty
   * @since Twenty Twenty 1.0
   */

get_header();
?>

<main id="site-content" role="main">
   <div class="post-inner thin entry-content">

      <?php

      // Set the Current Author Variable $curauth
      $curauth = (isset($_GET[,author_name'])) ? get_user_by(,slug', $author_name)
: get_userdata(intval($author));
      ?>

      <div class="author-profile">
         <h1>über <?php echo $curauth->nickname; ?></h1>
```

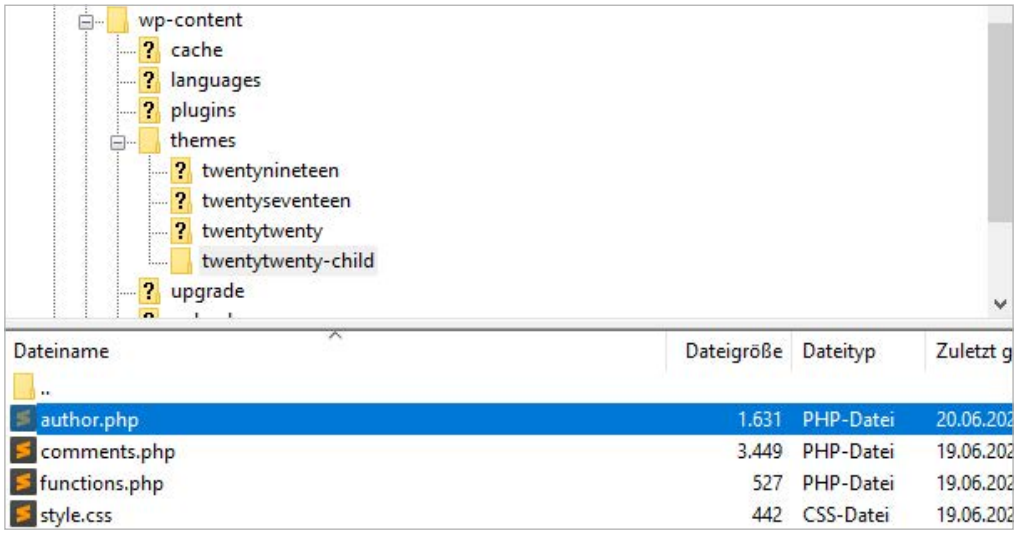
```
        <div class="author-photo">
        <?php echo get_avatar( $curauth->user_email , ,90 ,); ?>
        </div>
        <p>
        <?php echo $curauth->user_description; ?><br />
           <strong>Website: </strong><a href="<?php echo $curauth-
>user_url; ?>"><?php echo $curauth->user_url; ?></a><br />
        <strong>Kontakt: </strong><a href="mailto: <?php echo
get_the_author_meta(,user_email', 1); ?>">E-Mail schreiben</a>
        </p>

    </div>

    <h3>Beiträge von <?php echo $curauth->nickname; ?>:</h3>

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

        <h3>
        <a href="<?php the_permalink() ?>" rel="bookmark" title="Permanent Link:
        <?php the_title(); ?>">
        <?php the_title(); ?></a>
        </h3>

        <?php the_excerpt(); ?>

        <?php endwhile;

        // Previous/next page navigation.
        the_posts_pagination();

           else: ?>
           <p><?php _e(,No posts by this author.'); ?></p>

    <?php endif; ?>

</div>

<?php get_template_part( ,template-parts/footer-menus-widgets' ); ?>

<?php
get_footer();
```

Bei der Funktion **get_the_author_meta()** musst Du Deine Benutzer-ID eintragen (in unserem Beispiel: 1). Diese erfährst Du, wenn Du im Backend unter **Benutzer > Alle Benutzer** mit dem Mauszeiger über den Namen fährst. In der angezeigte URL ist dann „user_id=“ zu sehen – wenn nicht, ist es die Nummer 1.

Wichtig: Mit dem oben gezeigten Code wird die E-Mail-Adresse Deines Benutzernamens öffentlich. Wenn Du das nicht willst, entferne die folgende Zeile:

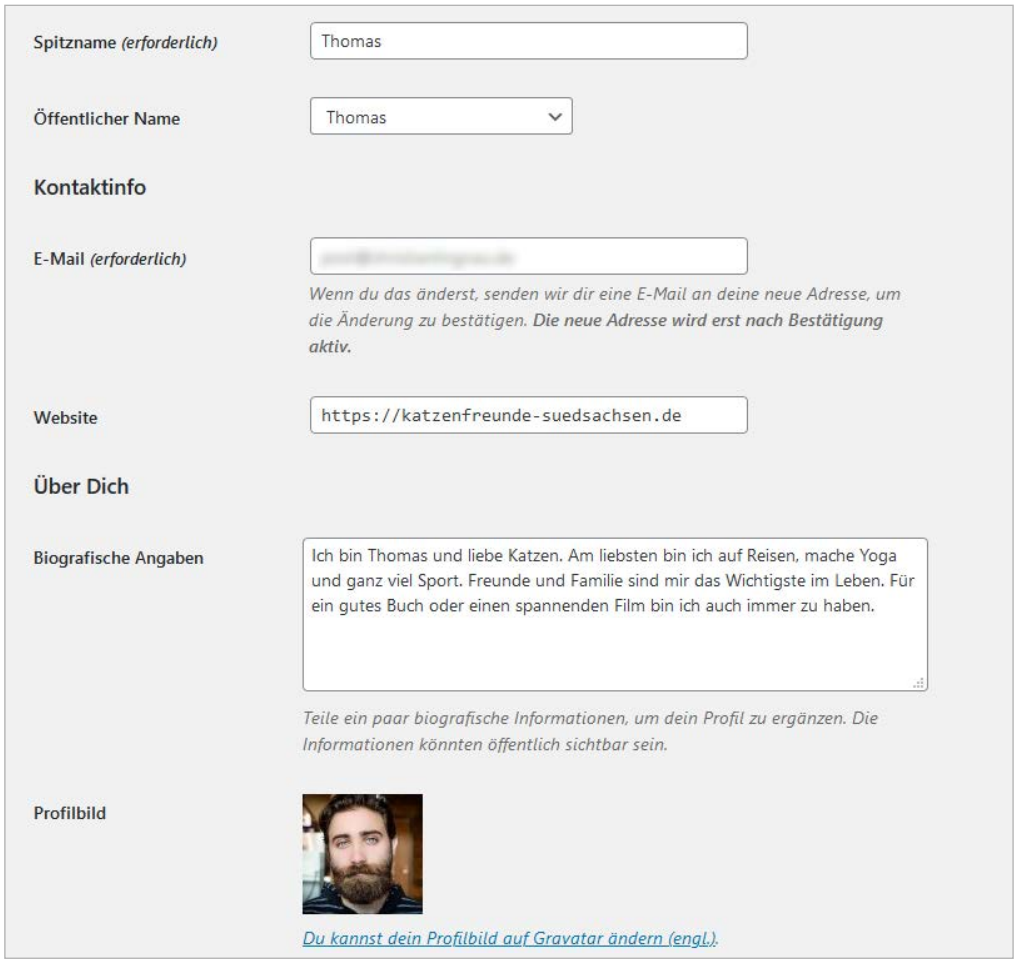
```
<strong>Kontakt: </strong><a href="mailto: <?php echo get_the_author_
meta(‚user_email‘, 1); ?>">E-Mail schreiben</a>
```

Die **author.php** lädst Du anschließend in das Hauptverzeichnis des Child-Themes:



2. Profil im Backend ausfüllen

Das Template kann Profilbild, biografische Angaben und Kontaktinfos nur anzeigen, wenn Du diese im Backend eingetragen hast (**Benutzer > Dein Profil**). Als Profilbild verwendet WordPress das bei **›Gravatar** hochgeladene Foto. Damit das funktioniert, musst Du Dich dort mit der gleichen E-Mail-Adresse wie im Blog anmelden und ein Bild einstellen.



Für ein vollständiges Autorenprofil brauchst Du nur die Felder im Backend auszufüllen.

3. CSS anpassen

Per CSS kannst Du anschließend Farben und Abstände anpassen. Die Anweisungen schreibst Du in die **style.css** Deines Child-Themes oder in den Customizer (**Design > Customizer > Zusätzliches CSS**). In unserem Beispiel färben wir den Hintergrund weiß, fügen einen einseitigen Rahmen hinzu (wie bei den Kommentaren) und sorgen für ein rundes Autorenbild.

```
.author-profile {
   background: #ffffff;
   padding: 1em;
      margin-right: auto;
         margin-left: auto;
```

```
        width: 100%;
        border-right: #cd2653 solid 10px;
}

.author-profile h1 {
   font-size: 3em;
      margin: 0em;
}

.author-profile p {
      margin-top: 1em;
}

.author-photo .avatar {
   float: left;
   text-align: left;
   margin-right: 1em;
      margin-top: 1em;
         border-radius: 50%;
}
```

Fertig! Links siehst Du die originale Archiv-Seite, rechts die angepasste Variante mit Profil:

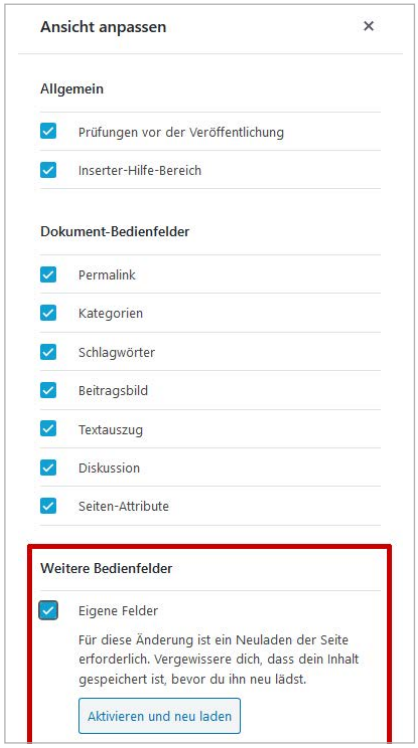
2.4 Benutzerdefinierte Felder anlegen

Vor allem spezialisierte Blogs haben bestimmte Datentypen. In einem Kochblog beispielsweise sind das Zutaten, Zubereitungszeit und Schwierigkeitsstufe. Mit den benutzerdefinierten Feldern musst Du diese Angaben nicht mehr im Fließtext unterbringen. Stattdessen legst Du individuelle Felder an, die ober- oder unterhalb des Beitrags angezeigt werden. Damit sparst Du Zeit und es sieht professioneller aus, wenn Du die Daten optisch vom Fließtext abgrenzt. Damit Suchmaschinen diese Inhalte besser verarbeiten, kannst Du Deine Daten auf Basis von **›schema.org** strukturieren oder ein **›Plugin** verwenden (mehr dazu in **›Kapitel 7.1**).

Wie Du die Funktion der benutzerdefinierten Felder nutzt, zeigen wir Dir in vier Schritten:

1. Ansicht anpassen

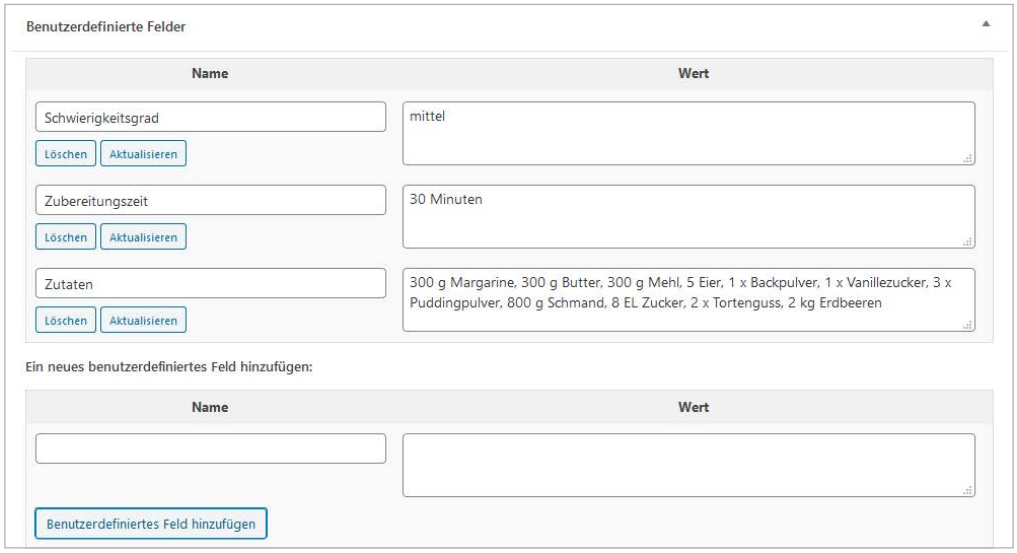
Benutzerdefinierte Felder werden im Backend auf der Beitragsebene unterhalb des Editors angelegt (**Beiträge > Alle Beiträge > Bearbeiten**). Falls Du diese Option nicht findest, musst Du über das Drei-Punkt-Menü oben rechts die Option **Ansicht anpassen** wählen. Setze dann ein Häkchen bei **Eigene Felder**.



Über das Menü oben rechts passt Du die Ansicht an.

2. Benutzerdefinierte Felder anlegen

Um beim Beispiel des Kochblogs zu bleiben: Während Du die Zubereitung im Fließtext beschreibst, legst Du nun unterhalb des Editors für Zutaten, Zubereitungszeit und Schwierigkeitsgrad einfach entsprechende benutzerdefinierte Felder an.
Als **Name** trägst Du den Datentyp (z. B. »Schwierigkeitsgrad«) und daneben den für dieses Rezept gültigen **Wert** ein (z. B. »mittel«).



Übrigens: Die Namen werden dabei gespeichert und stehen per Dropdown-Menü auch für andere Beiträge zur Verfügung. Das spart Zeit und stellt sicher, dass die Datentypen in den Beiträgen einheitlich benannt werden.

3. Felder im Beitrag anzeigen

Die benutzerdefinierten Felder zeigt WordPress erst mit der Funktion

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

an. Dazu fügst Du den Code innerhalb des **›Loops** ein. Der Loop bzw. die »Schleife« ist ein PHP-Code, mit dem WordPress die Inhalte anzeigt.

Wo die Funktion genau hingehört, hängt vom verwendeten Theme ab. Themes sind in der Regel kommentiert (z. B. »/* Start the Loop */«), sodass Du nicht lange suchen musst. Bei Twenty Twenty zum Beispiel bzw. dessen Child-Theme kannst Du die

Funktion in die Datei **content.php** oberhalb der Funktion **edit_post_link()**; einfügen. Am besten führst Du bei der Gelegenheit gleich eine neue Klasse wie »custom-fields« ein, mit der Du die Felder anschließend per CSS gestalten kannst.

```
<div class="section-inner">
    <?php
    wp_link_pages(
        array(
            'before'      => '<nav class="post-nav-links bg-light-background" aria-label="'
            'after'       => '</nav>',
            'link_before' => '<span class="page-number">',
            'link_after'  => '</span>',
        )
    );

    ?>

    <div class="custom-fields"><?php the_meta(); ?></div>

    <?
    edit_post_link();


    // Single bottom post meta.
    twentytwenty_the_post_meta( get_the_ID(), 'single-bottom' );

    if ( post_type_supports( get_post_type( get_the_ID() ), 'author' ) && is_single() ) {

        get_template_part( 'template-parts/entry-author-bio' );
```

Wichtig ist die korrekte Syntax: Hier musste oberhalb des markierten Bereichs ein schließendes Tag (?>) und unterhalb ein öffnendes Tag (<?) eingefügt werden. Zwischen den div-tags sollten keine Leerzeichen sein.

Beachte dabei die Verzeichnisstruktur: Die **content.php** liegt im Unterverzeichnis »template-parts«. Im Child-Theme muss die Hierarchie identisch sein. Erstelle daher einen gleichnamigen Ordner in Deinem Child-Theme und speichere dort die angepasste Kopie der **content.php**.

Sobald Du den Beitrag neu lädst, findest Du die benutzerdefinierten Felder unterhalb des Beitrags:

4. Daten vom Fließtext abgrenzen

Sofern im Theme nichts definiert wurde, wirken die Felder vielleicht etwas deplatziert. Um das zu ändern, kannst Du der **style.css** Deines Child-Themes zum Beispiel folgende Zeilen hinzufügen:

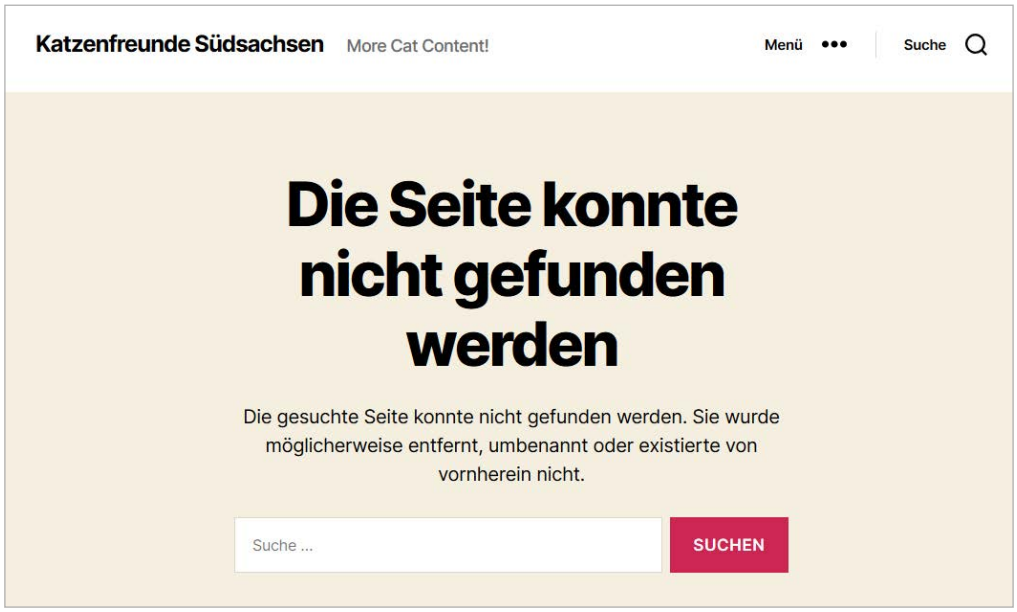
```
/* Custom Fields */
.custom-fields {
margin-top: 2em;
background-color: #ffffff;
padding: 0.5em;
margin-right: auto;
margin-left: auto;
max-width: 58rem;
width: 100%;
}

.custom-fields .post-meta-key {
font-weight: bold;
font-variant:small-caps;
list-style: none;
}

.custom-fields:empty {
   display: none;
}
```

Der pseudo-Klassen-Selektor **:empty** sorgt dafür, dass der Bereich nur bei vorhandenem Inhalt angezeigt wird. Fertig:

2.5 404-Fehlerseite individuell gestalten

Ob gelöschte Datei oder fehlerhafte URL: Der 404-Error hat verschiedene Ursachen. Wie Du das Problem löst, erfährst Du in **›Kapitel 5.3**. In manchen Fällen lässt sich der Fehler jedoch nicht vermeiden – etwa bei einem fehlerhaften Link oder wenn die URL von einem Nutzer falsch eingegeben wurde. WordPress gibt dann standardmäßig folgende Meldung aus: »Die Seite konnte nicht gefunden werden.«

Diese Meldung ist leider nicht besonders ansprechend. Allerdings wirst Du derartige 404-Templates in den meisten Themes vorfinden. Bei einer professionellen Fehlerseite mit individuellem Text, Bild und Widgets bleiben Deine Besucher eher auf Deiner Website.



Standard-404-Fehlerseite beim Theme Twenty Twenty

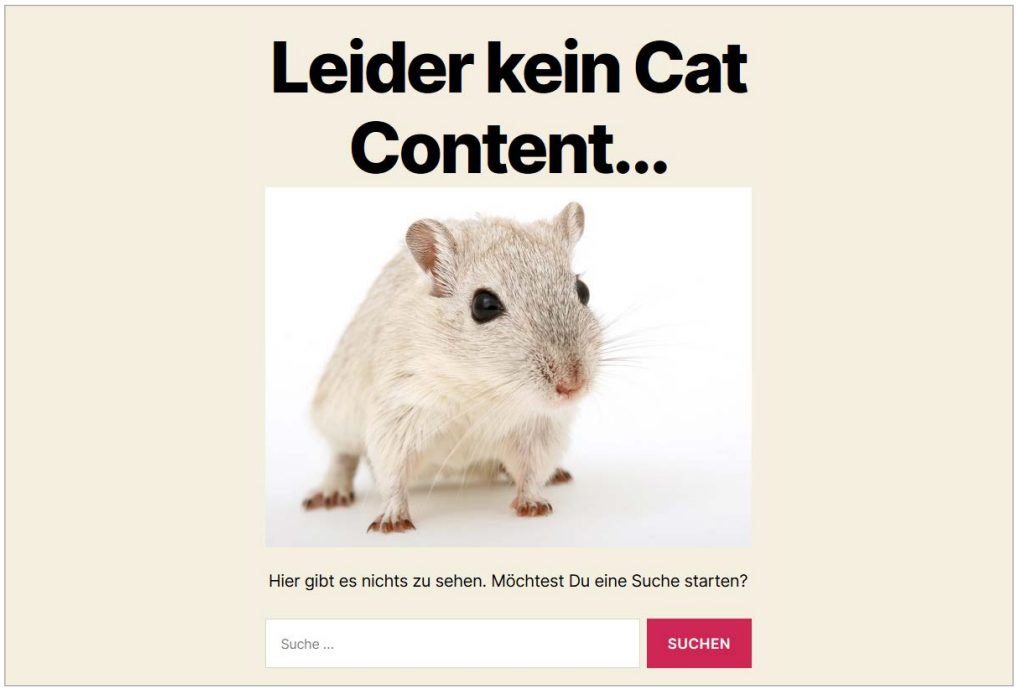
Bestehendes 404-Template anpassen

Das Template liegt in der Regel als **404.php** im Hauptverzeichnis des Themes. Um bei Twenty Twenty den Text zu ändern und ein Bild einzufügen, gehst Du wie folgt vor:

1. Ersetze innerhalb des h1-tags die Überschrift »Page Not Found« durch beispielsweise »Leider kein Cat Content…«. In der Zeile darunter löschst Du »The page you were looking for…« und gibst Deinen Text ein, etwa: »Hier gibt es nichts zu sehen. Möchtest Du eine Suche starten?«

2. Lade nun ein passendes Bild in die Mediathek. Wenn Du nach dem Upload auf das Bild klickst, wird die URL angezeigt. Füge diese per img-tag in eine beliebige Stelle des Templates ein. Bei unserem Katzenblog sieht das so aus:

```
<img src="https://katzenfreunde-suedsachsen.de/wp-content/
uploads/2020/06/animal-1238239_1280.jpg" alt="Maus" />
```



Mouse Content

404-Template erstellen

Falls Dein Theme über keine **404.php** verfügt, erstellst Du diese Datei selbst. Dazu kannst Du folgenden Code als Ausgangsbasis verwenden. Ersetze »twentytwenty« mit dem Namen Deines Themes und verwende die CSS-Klassen Deines Themes, damit das Layout zum Rest Deiner Website passt. Mehr Infos zu diesem Thema findest Du **›hier**.

```
<?php
/**
* The template for displaying 404 pages (Not Found)
*
* @package WordPress
* @subpackage Twenty_Twenty
* @since Twenty Twenty 1.0
*/

get_header(); ?>

   <div id="primary" class="content-area">
      <div id="content" class="site-content" role="main">

         <header class="page-header">
            <h1 class="page-title"><?php _e( ‚Not Found', ‚twentytwenty' ); ?></h1>
         </header>

         <div class="page-wrapper">
            <div class="page-content">
               <h2><?php _e( ‚This is somewhat embarrassing, isn't it?', ‚twentytwenty'
               ); ?></h2>
               <p><?php _e( ‚It looks like nothing was found at this location. Maybe try
               a search?', ‚twentytwenty' ); ?></p>

               <?php get_search_form(); ?>
            </div><!-- .page-content -->
         </div><!-- .page-wrapper -->

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

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

Widgets hinzufügen

Um Deine Besucher bei der Stange zu halten, kannst Du zusätzlich Widgets hinzufügen. Hier eignet sich am ehesten die Anzeige Deiner aktuellen Beiträge. So können die Besucher direkt zu den Inhalten springen.

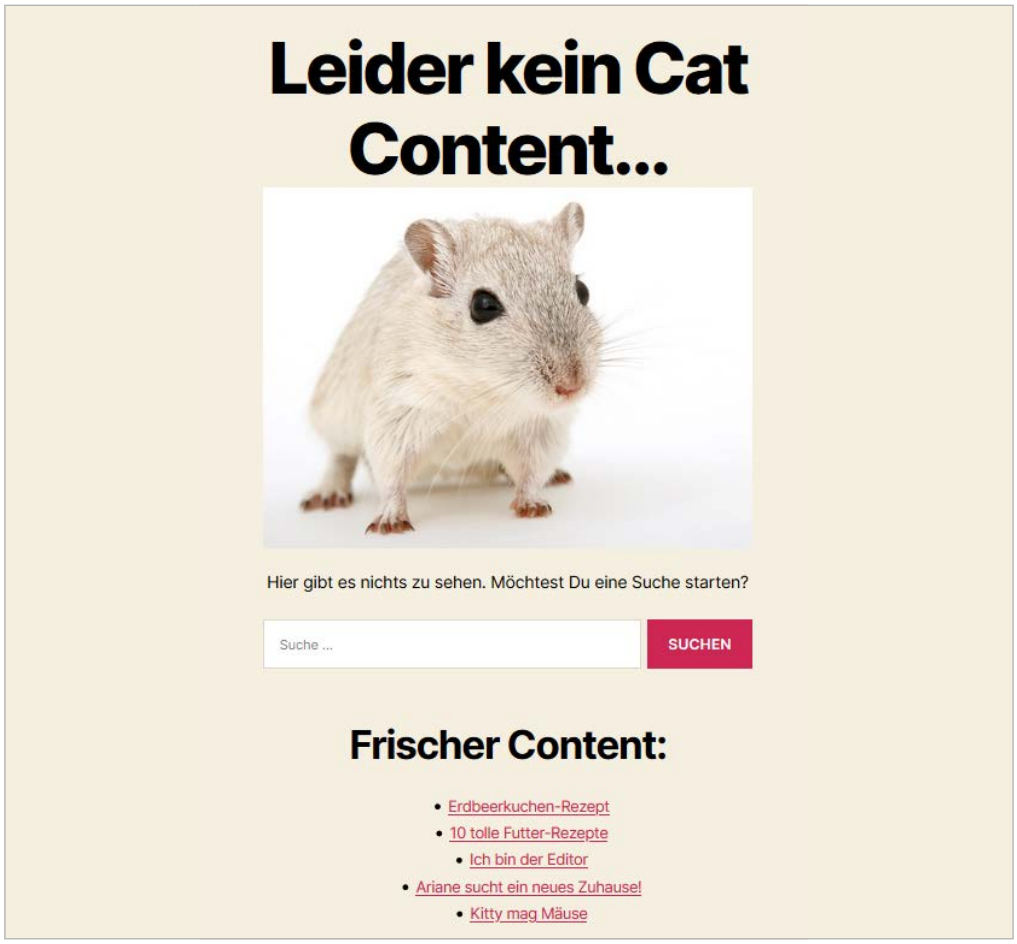
Gib den nachfolgenden Code in die **functions.php** Deines Child-Themes ein, um einen Widget-Bereich für die 404-Fehlerseite zu registrieren (in diesem Beispiel für ein Twenty-Twenty-Child-Theme):

```
//404-Widget-Bereich
function twentytwenty_child_widgets_init() {
register_sidebar(array(
‚name‘                    => ‚404-Fehlerseite‘,
‚id‘                      => ‚404-fehlerseite‘,
‚description‘            => ‚Widgets hier hinzufügen, um sie auf der 404-Fehlerseite
anzuzeigen.‘,
‚before_widget‘           => ‚<div class=”404-widget”>‘,
‚after_widget‘            => ‚</div>‘,
‚before_title‘            => ‚<h2>‘,
‚after_title‘             => ‚</h2>‘,
) );
}
add_action( ‚widgets_init‘, ‚twentyseventeen_child_widgets_init‘ );
```

Kopiere anschließend die folgenden Zeilen in die **404.php** des Child-Themes, um WordPress mitzuteilen, wo der Widget-Bereich anzeigt werden soll:

```
<!-- 404 Widget -->
    <?php if ( is_active_sidebar( ‚404-fehlerseite‘ ) ) : ?>
      <?php dynamic_sidebar( ‚404-fehlerseite‘ ); ?>
    <?php endif; ?>
```

Nun erscheint im Backend unter **Design > Widgets** der neue Bereich »404-Fehlerseite«. Ziehe dort wie gewohnt per Drag & Drop ein oder mehrere Widgets hinein. Mit den aktuellen Beiträgen sieht das anschließend so aus:



404-Fehlerseiten dürfen lustig sein

Wie Deine Besucher auf eine nicht gefundene Seite reagieren, hängt von Deiner Kreativität ab. Weil der 404-Fehler nicht dramatisch ist, dürfen 404-Meldungen **›auch lustig sein**. Solange der Error eine Ausnahme bleibt und Du Deinen Besuchern Optionen anbietest, bleiben sie Dir höchstwahrscheinlich erhalten.

Über die gezeigten Schritte hinaus kannst Du Dein Template natürlich weiter anpassen – **›vom Entfernen der Listenpunkte bis hin zum Full-width-Layout**. Vielleicht sehen Deine Besucher den Error dann ebenfalls mit Humor. Und wenn nicht, leitest Du sie einfach **›per Plugin** auf die Startseite weiter.

2.6 Landing Page mit einem Seiten-Template erstellen

Auf einer Landing Page sollen Besucher ohne Umwege und Ablenkung ein bestimmtes Produkt kaufen oder sich zu einer Veranstaltung anmelden können. Um dieses Ziel zu erreichen, muss eine Landing Page zielgruppenorientiert, argumentativ und auf das Wesentliche reduziert sein. Mit dem Klick auf den sogenannten Call-to-Action-Button wird aus dem Besucher schließlich ein Kunde.

WordPress hat mit den Seiten-Templates die passende Funktion bereits an Bord. In dieser Anleitung erfährst Du in vier Schritten, wie Du ein Seiten-Template für das Theme Twenty Twenty bzw. dessen Child-Theme anlegst. Dazu bauen wir als Beispiel eine einfache Landing Page, über die sich Besucher in unserem Beispiel für eine Yoga-Veranstaltung anmelden können.

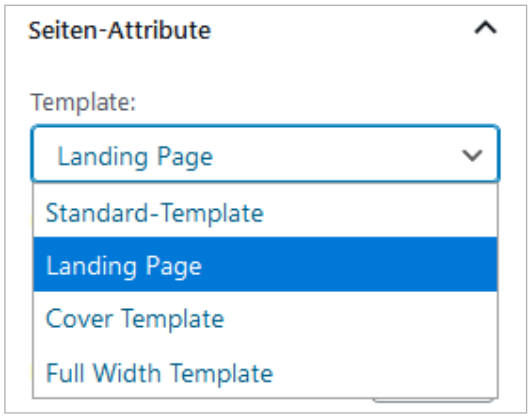
Schritt 1: Seiten-Template erstellen

Um das Template zu erstellen, lade die Datei **singular.php** aus dem Verzeichnis Deines Parent-Themes (**/wp-content/themes/twentytwenty**) herunter und benenne sie in **landing-page.php** um. Um daraus ein Seiten-Template zu machen, ersetzt Du den Kommentar am Anfang durch folgenden:

```
/**
 * Template Name: Landing Page

 *
 * @package Twenty Twenty Child
 */
```

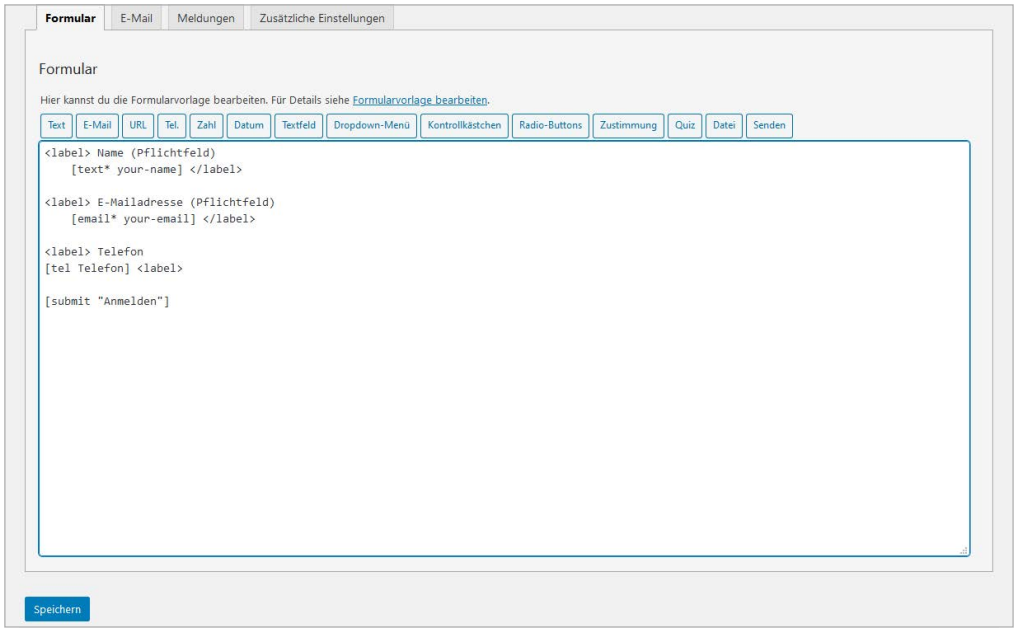
Anschließend lädst Du die Datei in das Hauptverzeichnis Deines Child-Themes.

Für die Landing Page erstellst Du nun eine Seite und präsentierst dort Dein Angebot. In der Seiten-Ansicht im Backend befindet sich rechts unter dem Reiter **Dokument** der Bereich **Seiten-Attribute**. Dort wechselst Du im Dropdown-Menü vom **Standard-Template** auf **Landing Page**.



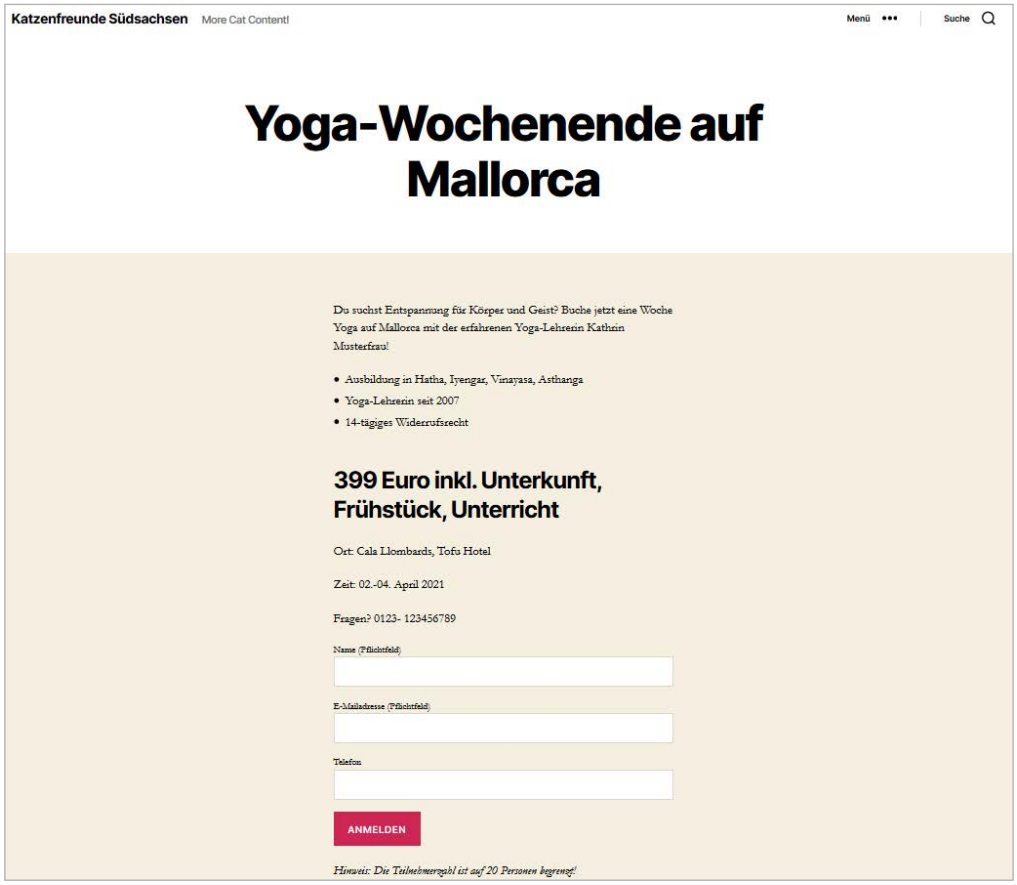
Schritt 2: Kontaktformular einfügen

Damit sich die Besucher für die Veranstaltung anmelden können, benötigst Du ein Kontaktformular. Empfehlenswert ist das **›Plugin Contact Form 7**, da es laufend gepflegt wird und sich flexibel anpassen lässt. Nach der Installation kannst Du im Backend unter **Formulare** das Standard-Formular bearbeiten. Nicht benötigte Labels kannst Du im Editor löschen und über die Buttons weitere Felder hinzufügen. In unserem Beispiel reichen Name, E-Mail-Adresse und Telefon. Den »Senden«-Button solltest Du so umbenennen, dass er zu Deinem Angebot passt (z. B. »Anmelden«, »Kaufen«, »Download«).



Um den Button des Formulars umzubenennen, änderst Du das Tag, z. B. [submit »Anmelden«].

Nach dem Speichern kopierst Du den blau markierten Shortcode oberhalb des Editors in die Landing Page. Diese sieht in unserem Fall anschließend so aus:



Ohne CSS-Anpassungen: Eine typische WordPress-Seite mit Header und Navigation.

Schritt 3: CSS anpassen

Mit dem nachfolgenden CSS-Code sieht das Ganze gleich viel besser aus. Das sind die wichtigsten Anpassungen im Überblick:

- Header und Navigation ausblenden
- Hintergrundbild einfügen
- Button-Farbe und -größe anpassen
- Schriftfarben ändern
- Link zum Impressum im Footer rechts platzieren

WordPress vergibt für Seiten-Templates neue Klassen nach dem Schema **body.page-template-TEMPLATENAME**. Damit kannst Du die Bereiche flexibel anpassen, ohne die anderen Seiten Deiner Website zu verändern:

```
/* Anpassungen Landing Page */

/* Header ausblenden */
body.page-template-landing-page .header-inner { display: none;
}

/* Hintergrundbild einfügen */ body.page-template-landing-page {
background-image: url(„/wp-content/uploads/2020/06/balance-110850_1280.jpg“);
background-repeat: no-repeat;
background-position: center;
background-size: cover;
}

/* Button */
body.page-template-landing-page .wpcf7-submit { width: 100%;
background: #9ACD32;
padding: 1rem;
}

/* Schriftfarben */
body.page-template-landing-page .entry-content p, h3, ul {
   color: #ffffff;
}
body.page-template-landing-page h1 {
   color: #282828;
}

/* Abstand nach oben reduzieren */
body.page-template-landing-page .post-inner.thin {
   padding-top: 0em;
}

/* Impressum-Link Farbe */
body.page-template-landing-page #site-footer a {
   color: #282828;
}
```

```
/* Impressum-Link rechts */
body.page-template-landing-page .landing-page-impressum {
   float: none;
   position: absolute;
   right: 10%;
}
```

Schritt 4: Impressum einfügen

Eine Landing Page muss – wie auch alle anderen Seiten einer Website – einen Link zum Impressum enthalten – am besten im Footer. Alles andere wird hier entfernt, damit Besucher nicht abgelenkt werden und aussteigen.

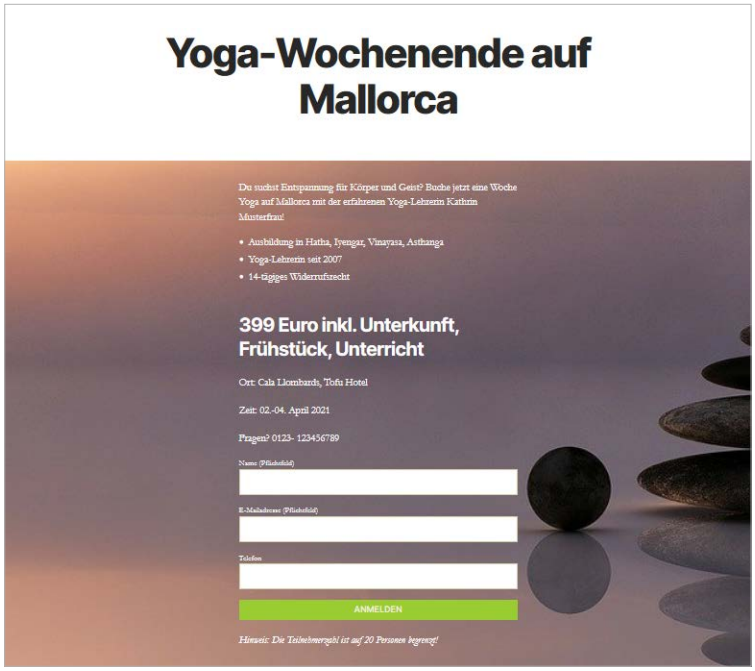
Bei Twenty Twenty ersetzt Du dazu einfach in der **landing-page.php** die Funktion <?php get_footer(); ?> durch Folgendes:

```
<?php get_template_part( ‚template-parts/footer-menus-widgets‘ ); ?>

<footer id=“site-footer“>

<a href=“https://DEINE-URL/impressum“ class=“landing-page-
impressum“>Impressum</a>

   </div>
```

Damit erhält die Landing Page einen Footer mit einem Link zum Impressum. Dessen Farbe und Platzierung haben wir über die Klasse »landing-page-impressum« bereits definiert.

Die Verweise zur Datenschutz-Seite, den AGB oder der Widerrufsbelehrung fügst Du wie das Impressum innerhalb der ID »site-footer« ein und vergibst dafür ggf. eigene Klassen. Über den Customizer kannst die CSS-Anweisungen live testen. Abschließend lädst Du die Dateien **landing-page.php** und **style.css** in das Hauptverzeichnis Deines Child-Themes.

Hier ist das Ergebnis unserer Yoga-Landing-Page:



Und so sieht das auf dem Smartphone aus:

2.7 Eigenes Theme entwickeln: Bootstrap-Navigation in WordPress implementieren

In **›Kapitel 1.3** haben wir beschrieben, wie Du ein Basis-Theme generierst und Bootstrap in WordPress grundsätzlich einbindest. Dieses Vorgehen funktioniert natürlich auch ohne lokale Testumgebung (was aber empfehlenswert ist) und mit anderen Themes. Darauf aufbauend zeigen wir Dir im Folgenden am Beispiel einer Navigationsleiste, wie Du Komponenten in WordPress einfügst.

Aus Sicherheitsgründen sollte mindestens ein alternatives Theme aktiviert sein, auf welches WordPress im Notfall zurückgreifen kann.

1. Vorbereitung

Zusammenfassend hier nochmal die in Kapitel 1.3 gezeigten Schritte:

1. Bootstrap in der aktuellen Version herunterladen und entpacken
2. Unterordner »bootstrap« in das Hauptverzeichnis des Themes laden
3. **functions.php** des Themes anpassen

2. Navigationsleiste einfügen

1. Um die **›Navbar** mit Dropdown-Menü in die Menü-Funktion von WordPress zu implementieren, lädst Du zunächst die **›class-wp-bootstrap-navwalker.php** von der GitHub Respository herunter und legst sie im Theme-Hauptverzeichnis ab.

2. Die **functions.php** muss nun erneut angepasst werden, damit sie die neue Datei aufrufen kann. Füge dazu folgende Zeile ganz ans Ende ein (verwende statt »meintheme« den Namen Deines Themes):

```
//**
 * Register Custom Navigation Walker
 */
function register_navwalker(){
    require_once get_template_directory() . ,/class-wp-bootstrap-navwalker.php‘;
}
add_action( ‚after_setup_theme‘, ‚register_navwalker‘ );
```

```
/**
 * Register Navigation Menu
 */
register_nav_menus( array(
   ‚primary‘ => __( ‚Primary Menu‘, ‚meintheme‘ ),
) );
```

3. Danach löschst Du die aktuelle WordPress-Navigation in der **header.php**, die sich ebenfalls im Theme-Ordner befindet. Entferne den kompletten Code zwischen

```
<header id=“masthead“ class=“site-header“>
```

und

```
</header><!-- #masthead -->
```

An gleicher Stelle fügst Du diese Zeilen ein:

```
<nav class=“navbar navbar-expand-md navbar-light bg-light“ role=“navigation“>
    <div class=“container“>
      <!-- Brand and toggle get grouped for better mobile display -->
      <button class=“navbar-toggler“ type=“button“ data-toggle=“collapse“
      data-target=“#bs-example-navbar-collapse-1“ aria-controls=“bs-example-
      navbar-collapse-1“ aria-expanded=“false“ aria-label=“Toggle navigation“>
        <span class=“navbar-toggler-icon“></span>
      </button>
      <a class=“navbar-brand“ href=“#“>Navbar</a>
        <?php
        wp_nav_menu( array(
          ‚theme_location‘    => ‚primary‘,
          ‚depth‘             => 2,
          ‚container‘ => ‚div‘,
          ‚container_class‘   => ‚collapse navbar-collapse‘,
          ‚container_id‘      => ‚bs-example-navbar-collapse-1‘,
          ‚menu_class‘        => ‚nav navbar-nav‘,
          ‚fallback_cb‘       => ‚WP_Bootstrap_Navwalker::fallback‘,
          ‚walker‘            => new WP_Bootstrap_Navwalker(),
        ) );
        ?>
      </div>
    </nav>
```

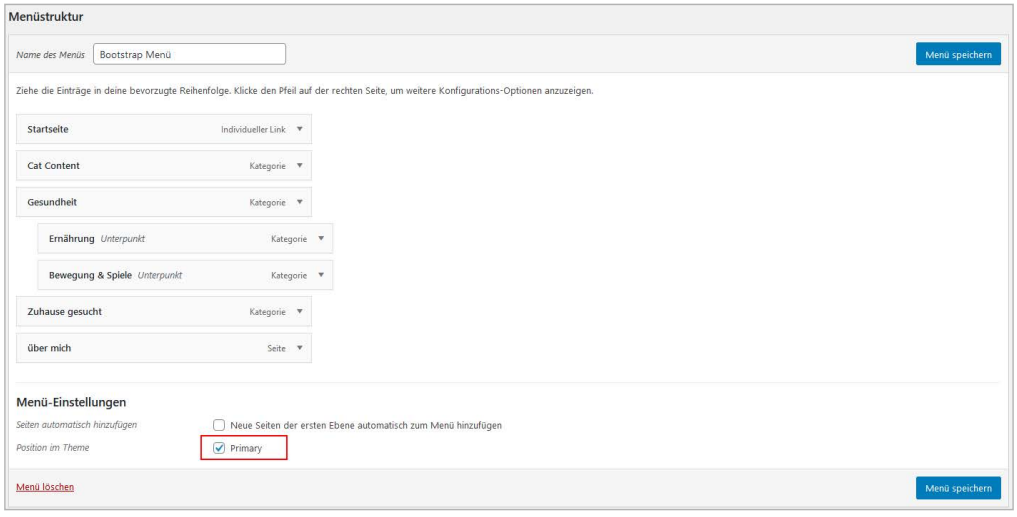
Der Bereich im Header sieht anschließend so aus:

```
<header id="masthead" class="site-header">

    <nav class="navbar navbar-expand-md navbar-light bg-light" role="navigation">
        <div class="container">
            <!--Brand and toggle get grouped for better mobile display-->
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#bs-example-
            navbar-collapse-1" aria-controls="bs-example-navbar-collapse-1" aria-expanded="false" aria-label="
            Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <a class="navbar-brand" href="https://katzenfreunde-suedsachsen.de">Katzenfreunde Südsachsen</a>
                <?php
                wp_nav_menu( array(
                    'theme_location'    => 'primary',
                    'depth'             => 2,
                    'container'         => 'div',
                    'container_class'   => 'collapse navbar-collapse',
                    'container_id'      => 'bs-example-navbar-collapse-1',
                    'menu_class'        => 'nav navbar-nav',
                    'fallback_cb'       => 'WP_Bootstrap_Navwalker::fallback',
                    'walker'            => new WP_Bootstrap_Navwalker(),
                ) );
                ?>
            </div>
        </nav>

</header><!-- #masthead -->
```

Anstelle von »Katzenfreunde Südsachsen« trägst Du als »navbar-brand« besser die URL und den Namen Deiner Website ein.

3. Menü erstellen

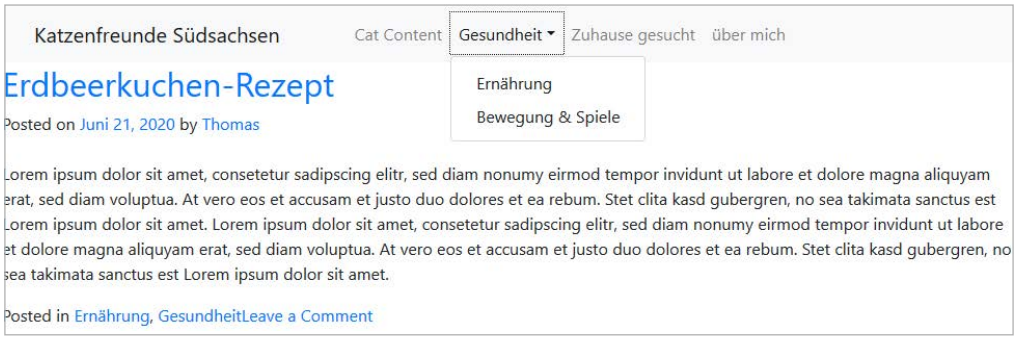
Wenn Du es bis hierhin geschafft hast, bist Du (fast) am Ziel angekommen! Nachdem Du die aktualisierten Dateien hochgeladen hast, kannst Du im Dashboard unter **Design > Menüs** ein Menü erstellen. Ggf. musst Du bestehende Menüs löschen. Damit ein Dropdown-Menü angezeigt wird, legst Du Unterpunkte an. Setze in den **Menü-Einstellungen** bei **Primary** ein Häkchen.



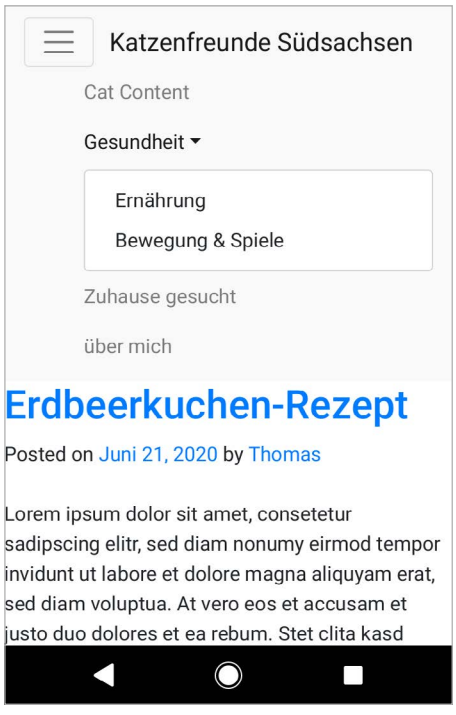
WordPress-Menü-Einstellungen: Per Drag & Drop kannst Du die Menüpunkte verschieben und so hierarchisch gliedern.

Fertig!

Das Ergebnis Deiner Arbeit kannst Du Dir im Frontend anschauen – ein Bootstrap-Menü im WordPress-Blog:



Desktop-Ansicht



Mobile Ansicht

2.8 Eigenes Theme entwickeln: Icons integrieren und Kommentarbereich optimieren

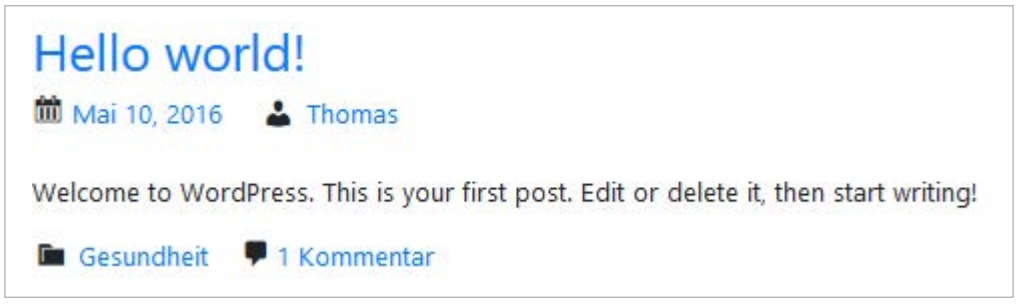
Im vorigen **›Kapitel 2.7** haben wir gezeigt, wie Du in das Basis-Theme Underscores eine Bootstrap-Navigationsleiste integrierst. Darauf aufbauend erfährst Du nun, wie Du Icons einbaust und Bootstrap-Kommentarfelder verwendest. Mit JavaScript kannst du obendrein die Eingaben Deiner Leser validieren und entsprechende Meldungen einblenden (z. B. »Bitte gib Deine E-Mail-Adresse ein.«).

1. Dashicons integrieren

Bootstrap 4 beinhaltet keine Icons. Gut, dass WordPress selbst welche an Bord hat: Die sogenannten **›Dashicons** sind die Icons, die Du bereits aus dem Backend kennst. Wenn Du folgende Funktion in die **functions.php** eingibst, kannst Du das Icon-Set auch für das Frontend nutzen:

```
// Enqueue Dashicons
function load_dashicons_front_end() { wp_enqueue_style( ‚dashicons‘ );
}
add_action( ‚wp_enqueue_scripts‘, ‚load_dashicons_front_end‘ );
```

Um das Theme grafisch aufzulockern, kannst Du nun die Links für Datum, Autor, Kategorien, Tags, Kommentare und Edit (bzw. Bearbeiten) mit passenden Icons versehen. Dazu musst Du die entsprechenden PHP-Dateien anpassen.



Vorher-Nachher: Mit Icons (rechts) wirkt das Theme professioneller.

Kalender-Icon

Zum Datum passt das Kalender-Dashicon. Trage zu diesem Zweck in die **content.php** (im Unterverzeichnis **template-parts**) unterhalb von

```
<div class="entry-meta">
```

Folgendes ein (Zeile 24):

```
<span class='dashicons dashicons-calendar'></span>
```

Weil das »Posted on« vor der Datumsangabe durch das Kalender-Icon nun überflüssig ist, kannst Du die Wörter in der Datei **/inc/template-tags.php** löschen (Zeile 30). Die Zeile sieht anschließend so aus:

```
Esc_html_x( ,%s', ,post date', ,meintheme' ),
```

Autor-Icon

Alle folgenden Anpassungen betreffen ebenfalls die **template-tags.php**. Um das User-Icon vor den Autorenamen zu platzieren, fügst Du (mit Leerzeichen am Anfang)

```
<span class='dashicons dashicons-admin-users'></span>
```

in die Zeile 50 ein:

```
echo ,<span class="byline"> <span class="dashicons dashicons-admin-users"></
span>' . $byline . ,</span>'; // phpcs:ignore WordPress.Security.EscapeOutput.
OutputNotEscaped
```

Das Wort »by« wird nicht mehr benötigt. Lösche es einfach (Zeile 46), sodass die Zeile wie folgt aussieht:

```
esc_html_x( , %s', ,post author', ,meintheme' ),
```

Kategorien-Icon

Die weiteren Meta-Angaben werden unterhalb des Inhalts angezeigt. Trotzdem bleibst Du in der gleichen Datei, um auch die Kategorien mit einem Dashicon zu versehen. Die entsprechende Klasse

```
<span class="dashicons dashicons-category"></span>
```

fügst Du in die Zeile 66 ein. Außerdem kannst Du wieder überflüssige Wörter streichen (»Posted in«):

```
printf( ‚<span class="cat-links"><span class="dashicons dashicons-category"></span>‘ . esc_html__( ‚Posted in %1$s‘, ‚meintheme‘ ) . ‚</span>‘, $categories_list ); // phpcs:ignore
```

Tags-Icon

Wie bei den Kategorien kannst Du überflüssigen Text (»Tagged«) entfernen (Zeile 73). Hier wird die Klasse verwendet:

```
<span class="dashicons dashicons-tag"></span>
```

```
printf( ‚<span class="tags-links"><span class="dashicons dashicons-tag"></span>‘ . esc_html__( , %1$s‘, ‚meintheme‘ ) . ‚</span>‘, $tags_list ); // phpcs:ignore WordPress.Security.EscapeOutput.OutputNotEscaped
```

Kommentare-Icon

In Zeile 77 fügst Du die Klasse

```
<span class="dashicons dashicons-admin-comments"></span>
```

hinzu, damit vor dem Kommentare-Link eine Sprechblase angezeigt wird:

```
echo ‚<span class="comments-link"><span class="dashicons dashicons-admin-comments"></span>‘;
```

Edit-Icon

Für ein Edit-Icon ergänzt Du die Zeile 109 um die Klasse

```
<span class="dashicons dashicons-edit"></span>
```

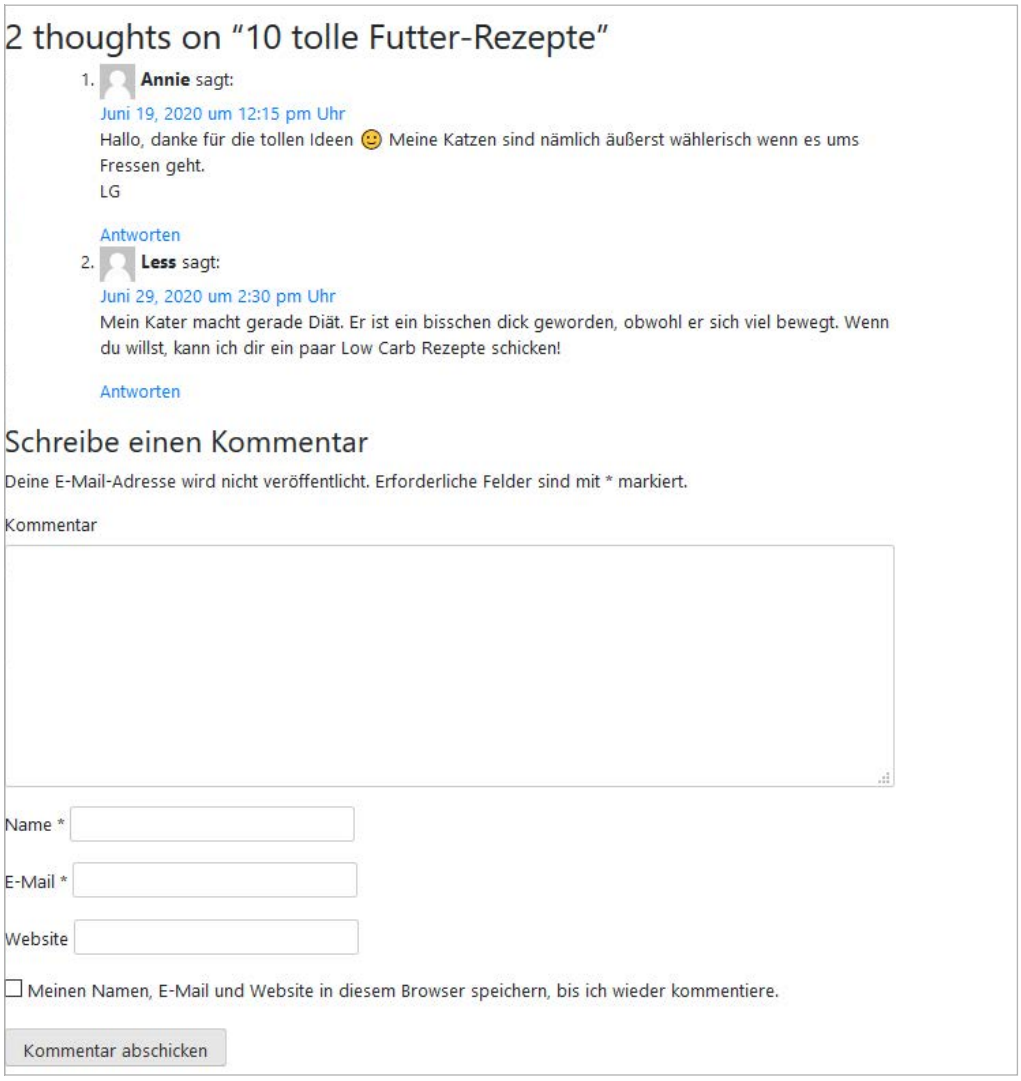
:

```
‚<span class="edit-link"><span class="dashicons dashicons-edit"></span>‘,
```

Zum Schluss kannst Du die Icons bequem über ihre Klassen in der **style.css** ausrichten:

```
*  Dashicons ausrichten */
.dashicons-category {
    margin: 0.1em;
}
.dashicons-tag, .dashicons-admin-comments, .dashicons-admin-users, .dashicons-edit {
    margin: 0.1em 0.1em 0.1em 1em;
}
```

2. Kommentarbereich mit Bootstrap optimieren



Der Underscores-Kommentarbereich sieht im Auslieferungszustand etwas blass aus.

Der Kommentarbereich ist ein gutes Beispiel für die Verwendung von Bootstrap-Komponenten. Die Eingabefelder von Bootstrap sehen nicht nur besser aus als Underscores-Original, sondern lassen sich per CSS auch einfacher anpassen.

Öffne dazu die **comments.php** und lösche diese Zeilen:

```
comment_form();
    ?>
```

Füge an gleicher Stelle den folgenden Code-Block ein:

```
/* Bootstrap Kommentarfelder */

    //$consent
    if ( has_action( ‚set_comment_cookies‘, ‚wp_set_comment_cookies‘ ) ) {
        $consent        = empty( $commenter[‚comment_author_email‘] ) ? ‚‘ : ‚
        checked=“checked“‘
        $fields[‚cookies‘] = ‚<p class=“comment-form-cookies-consent“><input
        id=“wp-comment-cookies-consent“ name=“wp-comment-cookies-consent“
        type=“checkbox“ value=“yes“‘ . $consent . ‚ />‘ .
. __( ‚Save my name, email, and website in this browser for the next time I
comment.‘ ) . ‚</label></p>‘;
        }

//Bootstrap Klassen
    ob_start();
    $commenter = wp_get_current_commenter();
    $req = true;
    $aria_req = ( $req ? „ aria-required=‘true‘“ : ‚‘ );

    $comments_arg = array(
        ‚form‘                           => array(
            ‚class‘                      => ‚form-horizontal‘
            ),
        ‚fields‘ => apply_filters( ‚comment_form_default_fields‘, array(
                ‚author‘                 => ‚<div class=“form-group“>‘ . ‚<label for=“author“>‘
                . __( ‚Name‘, ‚strato‘ ) . ‚</label> ‚ . ( $req ? ‚<span>*</span>‘ : ‚‘ ) .
            ‚<input id=“author“ name=“author“ class=“form-control“ type=“text“
            value=““ size=“30“‘ . $aria_req . ‚ />‘.
                ‚<p id=“v1“ class=“text-danger“></p>‘ . ‚</div>‘,
```

```
            ‚email‘                => ‚<div class="form-group">‘ .‘<label for="email">‘ .
            __( ‚E-Mail‘, ‚strato‘ ) . ‚</label> , . ( $req ? ‚<span>*</span>‘ : ‚‘ ) .
        ‚<input id="email" name="email" class="form-control" type="text" value=""
        size="30"‘ . $aria_req . ‚ />‘.
            ‚<p class="text-danger"></p>‘ . ‚</div>‘,
            ‚url‘                  => ‚<div class="form-group">‘ .‘<label for="url">‘ .
                                    __( ‚Website‘, ‚strato‘ ) . ‚</label> , .
                                                ‚<input id="url" name="url"
class="form-control" type="text" value="" size="30" />‘.
                                            ‚<p class="text-danger"></p>‘ . ‚</div>‘,
            ‚cookies‘              => ‚<div class="comment-form-cookies-
                                    consent"><input id="wp-comment-cookies-
                                    consent" name="wp-comment-cookies-consent"
                                    type="checkbox"‘ . $consent . ‚ />‘ .
            ‚<label for="wp-comment-cookies-consent">‘ . __( ‚Save my name, email,
            and website in this browser for the next time I comment.‘ ) . ‚</label></
            div>‘,)),
            ‚comment_field‘        => ‚<div class="form-group">‘ . ‚<label
                                    for="comment">‘ . __( ‚Kommentar‘, ‚strato‘ ) . ‚</
                                    label><span>*</span>‘ .
                                        ‚<textarea id="comment" class="form-
                                        control" name="comment" rows="3" aria-
                                        required="true"></textarea>
                                                <p id="v3"class="text-danger"></
                                                p>‘ . ‚</div>‘,
            ‚class_submit‘         => ‚btn btn-primary‘
        ); ?>
    <?php comment_form($comments_arg);
    echo str_replace(‚class="comment-form"‘,‘class="comment-form"
    name="commentForm" onsubmit="return validateForm();"‘,ob_get_clean());
    ?>
```

Um zum Antworten einen Bootstrap-Button zu verwenden, fügst Du diese Zeilen am Ende der **functions.php** ein:

```
// Bootstrap Submit Button
add_action(‚comment_form‘, ‚bootstrap4_comment_button‘ );
function bootstrap4_comment_button() {
    echo ‚<button class="btn btn-primary" type="submit">‘ . __( ‚Kommentar
    abschicken‘ ) . ‚</button>‘;
}
```

Nach der Aktualisierung erscheint unterhalb des grauen Kommentar-Buttons das blaue Bootstrap-Pendant. Über die **style.css** blendest Du den alten Button einfach aus:

```
/* Underscores-Button ausblenden */
.form-submit {
 display: none;
}
```

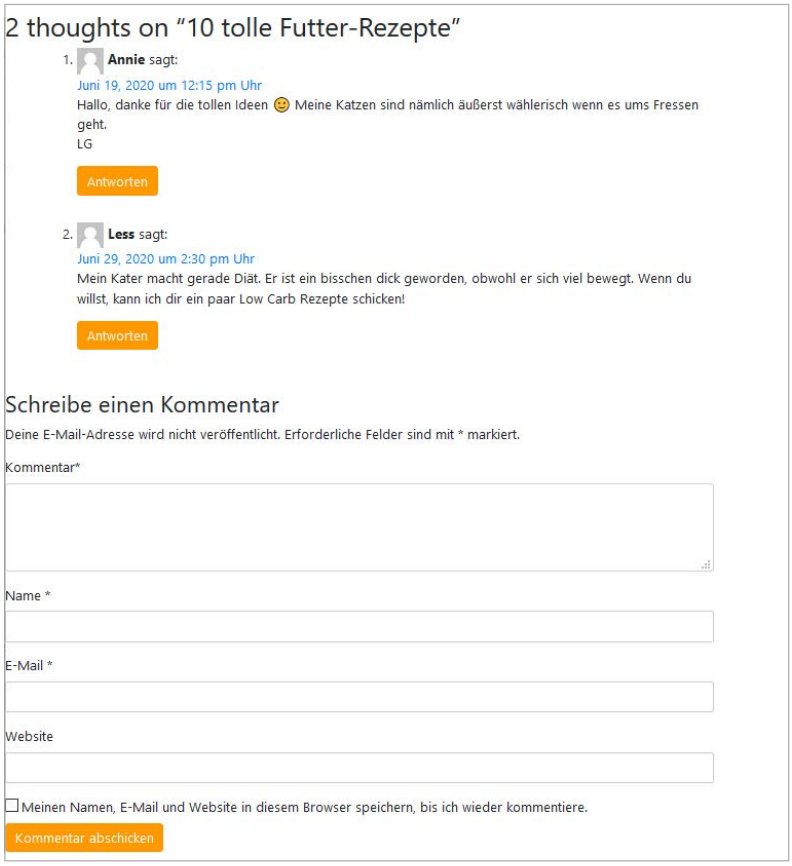
Damit auch die Reply-Links als Buttons erscheinen, ersetzt Du in der **functions.php** die originale Klasse durch die Bootstrap-Klasse (in diesem Fall **.btn-primary**):

```
// Bootstrap Antworten-Button
add_filter(‚comment_reply_link‘, ‚replace_reply_link_class‘);
function replace_reply_link_class($class){
    $class = str_replace(„class=‘comment-reply-link“, „class=‘btn btn-primary“,
    $class);
    return $class;
}
```

Über das Stylesheet kannst Du die Abstände und Buttons-Farben anpassen:

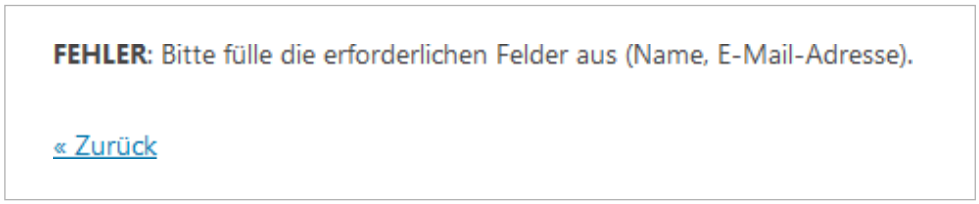
```
/* Abstand Reply-Links */
.reply {
    padding-bottom: 2rem;
}
/* Abstand Cookie-Checkbox */
input#wp-comment-cookies-consent {
    margin-right: 0.2rem;
}
/* Farbe Bootstrap-Buttons */
.btn.btn-primary {
        background-color: #ff9900;
        border: #ff9900;
        }
```

Wenn alles erledigt ist, sieht Dein Kommentarbereich so aus:



3. Validierung der Kommentarfelder

Füllst Du die Pflichtfelder im Kommentarbereich nicht aus, leitet Dich WordPress standardmäßig auf eine neue Seite mit folgender Nachricht weiter:



Wäre es nicht besser, auf der Seite zu bleiben und entsprechende Meldungen direkt bei den Kommentarfeldern anzuzeigen?

Falls Du Dir den neuen Code für die **comments.php** genau angeschaut hast, sind Dir vielleicht die IDs v1, v2 und v3 aufgefallen. Damit kannst Du die Kommentarfelder per JavaScript direkt beim Absenden eines Kommentars validieren und individuelle Meldungen anzeigen. Füge dazu diesen Code-Block in die **comments.php** direkt oberhalb von

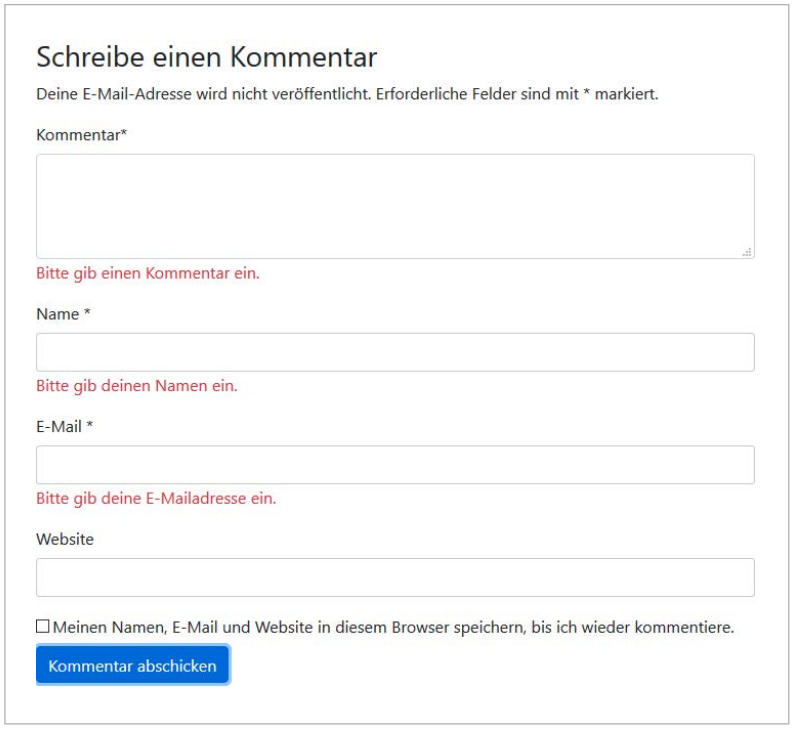
```
</div><!-- #comments -->
```

ein:

```
<script type="text/javascript">
    /* basic javascript form validation */
    function validateForm() {
    var form  =  document.forms["commentForm"];
    x  = form["author"].value,
    y  = form["email"].value,
    z  = form["comment"].value,
    flag = true,
    v1 = document.getElementById("v1"),
    v2 = document.getElementById("v2"),
    v3 = document.getElementById("v3");

  if (x == null || x == "") {
    v1.innerHTML = "<?php echo __(,Bitte gib deinen Namen ein.', ,strato'); ?>";
        xyz = false;
      } else {
        v1.innerHTML = "";
      }

      if (y == null || y == "") {
        v2.innerHTML = "<?php echo __(,Bitte gib deine E-Mail-Adresse ein.',
        ,strato'); ?>";
        xyz = false;
      } else {
        v2.innerHTML = "";
      }

      if (z == null || z == "") {
        v3.innerHTML = "<?php echo __(,Bitte gib einen Kommentar ein.', ,strato');
        ?>";
        xyz = false;
```

```
        } else {
            v3.innerHTML = „“;
        }

        if (xyz == false) {
            return false;
        }

    }
</script>
```

Wenn nun ein Besucher die Felder für Kommentar, Name oder E-Mail leer lässt und auf **Kommentar absenden** klickt, erhält er direkt folgende Meldungen:



2.9 Eigenes Theme entwickeln: Strukturieren mit HTML und CSS

Im vorigen ›**Kapitel 2.8** haben wir gezeigt, wie Du Icons und Bootstrap-Elemente für die Entwicklung Deines Themes auf Basis von Underscores verwendest. Unabhängig davon, ob es sich um ein eigenes oder vorgefertigtes Theme handelt: Um möglichst flexibel zu sein, hilft es, die HTML-Struktur anzupassen und dann per CSS zu gestalten. Auf diese Weise kannst Du etwa den Content-Bereich mittig platzieren oder eine Sidebar einfügen.

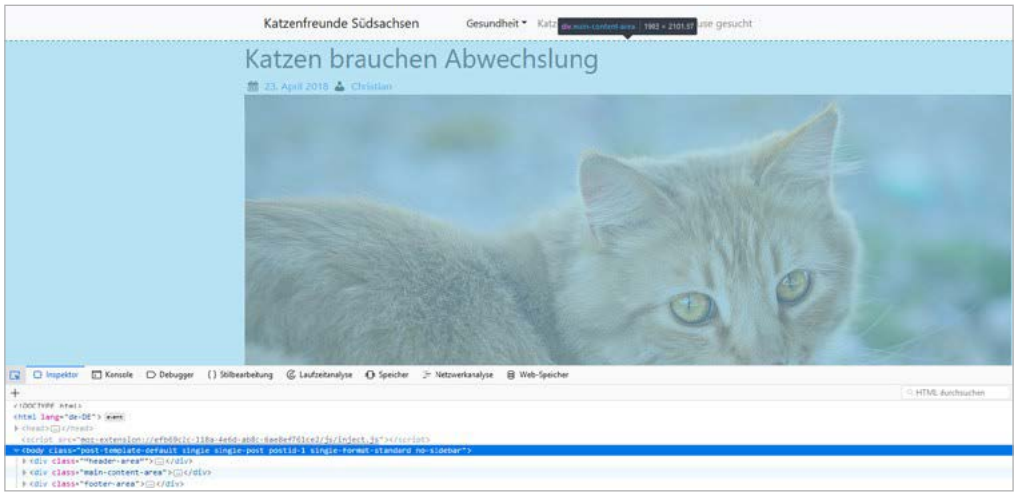
Flexibel ist besser

Underscores teilt den Inhaltsbereich in **#Primary** (Content) und **#Secondary** (Sidebar). Die Sidebar befindet sich standardmäßig unterhalb des Content-Bereichs, weil dieser die komplette Breite einnimmt. Um den linksbündigen Content-Bereich in die Mitte zu verschieben, könntest Du einfach folgende CSS-Anweisung verwenden:

```
/* Content mittig platzieren */
#page {
   width: 1170px;
   margin: 0 auto;
}
```

Das Problem dabei ist allerdings, dass der Content-Bereich auf die angegebene Breite beschränkt bleibt. Nicht möglich wäre daher beispielweise ein Header, der die volle Bildschirmbreite einnimmt. Darum ist es besser, auf den globalen **#page**-Container zu verzichten und stattdessen separate Klassen einzuführen.

Die Bereiche Header, Content und Footer bekommen je zwei Klassen: Eine eigene Klasse (**area**) für unabhängige CSS-Anpassungen und eine übergeordnete Klasse (**main-page**). Wir passen zu diesem Zweck drei Dateien an, die im Hauptverzeichnis des Themes liegen: **header.php**, **footer.php** und **style.css**.



Die neue Klasse »main-content-area« wird die volle Breite einnehmen.

HTML-Struktur anpassen

Der Container **#page** beginnt im Header und endet im Footer. Um ihn zu entfernen, löschst Du in der **header.php** die Zeile

```
<div id="page" class="site">
```

und in der **footer.php** folgenden Code:

```
</div><!-- #page -->
```

Header

Danach fügst Du in der **header.php** an gleicher Stelle eine Klasse für den Header (**.header-area**) und direkt darunter (also innerhalb des Containers) die zweite Klasse (**main-page**) für die gesamte Seite ein:

```
<div class="header-area">
  <div class="main-page">
```

Schließe die beiden Container unmittelbar nach

```
</header><!-- #masthead -->
```

mit

```
    </div>
</div>
```

Content

Das Gleiche wiederholst Du für den Content. Füge ebenfalls in der **header.php** ganz unten Folgendes ein:

```
<div class="main-content-area">
<div class="main-page">
```

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

Danach schließt Du die beiden div-tags in der **footer.php** oberhalb von

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

mit

```
</div>
    </div>
```

Footer

Aller guten Dinge sind drei. Gib nun für den Footer in der **footer.php** oberhalb von

```
<footer id="colophon" class="site-footer">
```

diese Zeilen ein:

```
div class="footer-area">
        <div class="main-page">
```

Die Tags werden unterhalb von

```
</footer><!-- #colophon -->
```

geschlossen mit

```
    </div>
</div>
```

Content mittig platzieren

Nun verfügt das Theme über je eine Klasse für Header, Content und Footer, die bei Bedarf die volle Breite nutzen können. Mit der übergeordneten Klasse **main-page** kannst Du in der **style.css** eine fixe Breite von beispielsweise 1.170 Pixeln definieren. Die Container **#Primary** und **#Secondary** verwenden unabhängig vom Inhalt immer die maximale Breite. So bleibt die Sidebar bzw. der Widget-Bereich unterhalb des (mittigen) Contents:

```
/* Breite 1170px */
.main-page {
    max-width: 1170px;
    margin: 0 auto;
}

#primary, #secondary {
    float: none;
    padding: 0;
    width: 100%;
}
```

Über **.main-content-area** kannst Du nun den Abstand des Contents zum Footer vergrößern:

```
/* Abstand Content-Footer*/
.main-content-area {
    padding-bottom: 2%;
}
```

Mittig platzierte Inhalte mit einer maximalen Breite von 1.170 Pixeln. Damit die Bilder nicht zu viel Platz einnehmen, ist ein Seitenverhältnis von 16:9 empfehlenswert (1170x658 Pixel).

Sidebar einfügen

Falls Du in Deinem Fullwidth-Layout doch die Sidebar vermisst, kannst Du problemlos zur bekannten Blog-Aufteilung wechseln. Dabei solltest Du allerdings die unterschiedlichen Auflösungen von Desktops und mobilen Geräten beachten. Die Bootstrap-Navigationsleiste wechselt bei einer Bildschirmbreite unterhalb von 768 Pixeln in den mobilen Modus. An diesem Breakpoint sollte auch die Sidebar nach unten wandern. Ändere dazu die Media Query in der **style.css** wie folgt:

```
/* Sidebar ausblenden (mobil) */
@media screen and (min-width: 768px) {
    .menu-toggle {
        display: none;
    }
    .main-navigation ul {
        display: block;
    }
```

```
    #primary {
    float: left;
    padding-right: 3%;
    width: 66.5%
    }

    #secondary {
    float: left;
    width: 33.5%
    }
}
```

Damit sagst Du Deinem Theme, dass das Bootstrap-Menü ab einer Bildschirmbreite von 768 Pixeln nicht mehr eingeklappt wird. Mit den zusätzlichen Anweisungen unten legst Du fest, dass der Content zwei Drittel und die Sidebar ein Drittel der Breite einnehmen.

Layout mit Sidebar in der Desktop-Ansicht

Falls Dich die Listenpunkte in den Widgets stören – weg damit:

```
.widget-area ul {
    list-style-type: none;
    margin: 0px;
    padding-left: 0px;
}
```

In der mobilen Ansicht wird das Menü eingeklappt und die Sidebar unterhalb des Inhaltsbereichs platziert:

Der Anfang ist gemacht

Ob Fullwidth- oder klassisches Blog-Layout mit Sidebar: Mit den gezeigten Modifikationen der HTML-Struktur hast Du die Gestaltung Deines Themes selbst in der Hand. Wenn Header oder Footer breiter sein sollen als der Inhaltsbereich, verwendest Du die separaten **area**-Klassen. Farben, Abstände, Schriften kannst Du nach Lust und Laune über die **main-page** ändern.

3 Plugins

3.1 Dein eigenes WordPress-Plugin leicht gemacht

Bestimmt hast Du auch schon nach Tricks gesucht, um einfache Funktionen in WordPress hinzuzufügen – beispielsweise Shortcodes, Code-Schnipsel zum Integrieren von Google Analytics oder ein Custom Post Type. Meist findest Du dabei Code-Schnipsel, die Du per Copy-and-Paste in die **functions.php** Deines WordPress-Themes übernimmst.

Die **functions.php** zu verändern, hat aber auch Nachteile:

- Änderst Du die **functions.php** direkt im Theme, gehen die Änderungen mit dem nächsten Update verloren. Du solltest dafür also ohnehin immer **›ein Child-Theme anlegen**.
- Wechselst Du eines Tages Dein Theme, musst Du die Änderungen in der **functions.php** mühselig heraussuchen und in das neue Theme übertragen.
- Je mehr Du in der **functions.php** ergänzt, desto unübersichtlicher wird die Datei und desto aufwändiger die Fehlersuche.

Die Lösung: Dein eigenes Plugin

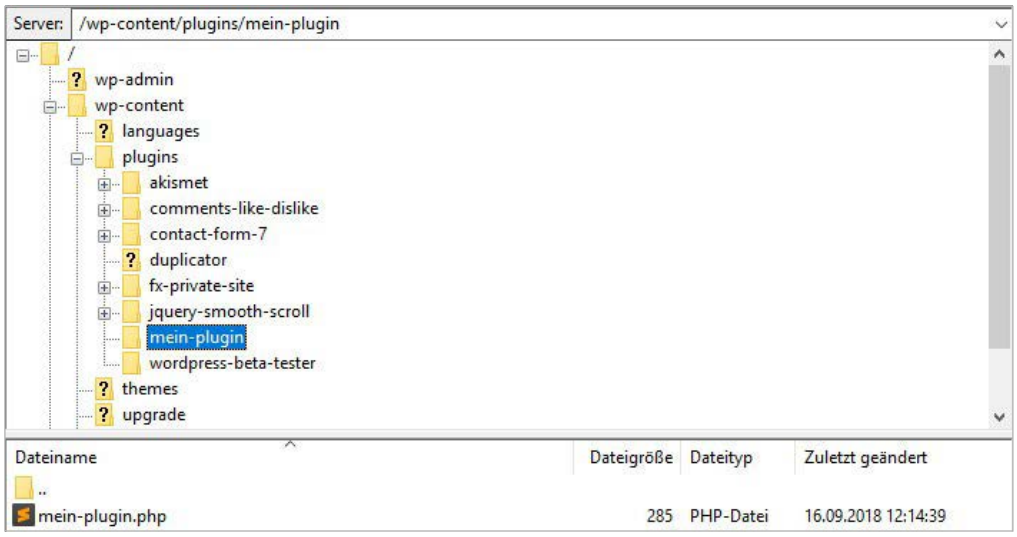
All diese Probleme vermeidest Du, indem Du die meisten Deiner Anpassungen und Ergänzungen stattdessen in ein eigenes WordPress-Plugin packst. Dort verwaltest Du Code-Schnipsel komfortabel. Wenn sich ein Fehler einschleicht, nimmst Du das Plugin kurzfristig offline und musst nicht am Livesystem in der **functions.php** auf Fehlersuche gehen. Auch in ein neues Theme kannst Du ein solches Plugin mitnehmen.

Ein Plugin zu erstellen, ist sehr einfach

Ein eigenes Plugin zu programmieren, klingt kompliziert, ist es aber nicht. Genau genommen brauchst Du erst einmal nur ein paar Zeilen Text in einer neu anzulegenden PHP-Datei.

```
<?php
defined( ‚ABSPATH‘ ) or die( ‚No script kiddies please!‘ );
/* also read https://codex.wordpress.org/Writing_a_Plugin */
/*
Plugin Name: Name Deines Plugins
Description: kurze Beschreibung
*/
/* Plugin-Code UNTERhalb dieser Zeile */
/* Plugin-Code OBERhalb dieser Zeile */
```

Speichere diese Datei mit einem beliebigen Dateinamen und der Dateiendung **.php** ab, lege im Plugin-Verzeichnis Deiner WordPress-Installation (**/wp-content/plugins**) einen neuen Ordner mit dem gleichen Namen an und lade die Datei dort hinein. Achte darauf, dass der Plugin-Name einzigartig ist und nicht denselben Namen trägt wie ein anderes Plugin.



Dein Plugin gehört in denselben Ordner wie alle anderen WordPress-Plugins.

Anschließend musst Du noch das eben neu erstellte Plugin in WordPress aktivieren. Jetzt hast Du Dein eigenes WordPress-Plugin – allerdings noch ohne Funktion.

Eigene Funktionalität hinzufügen

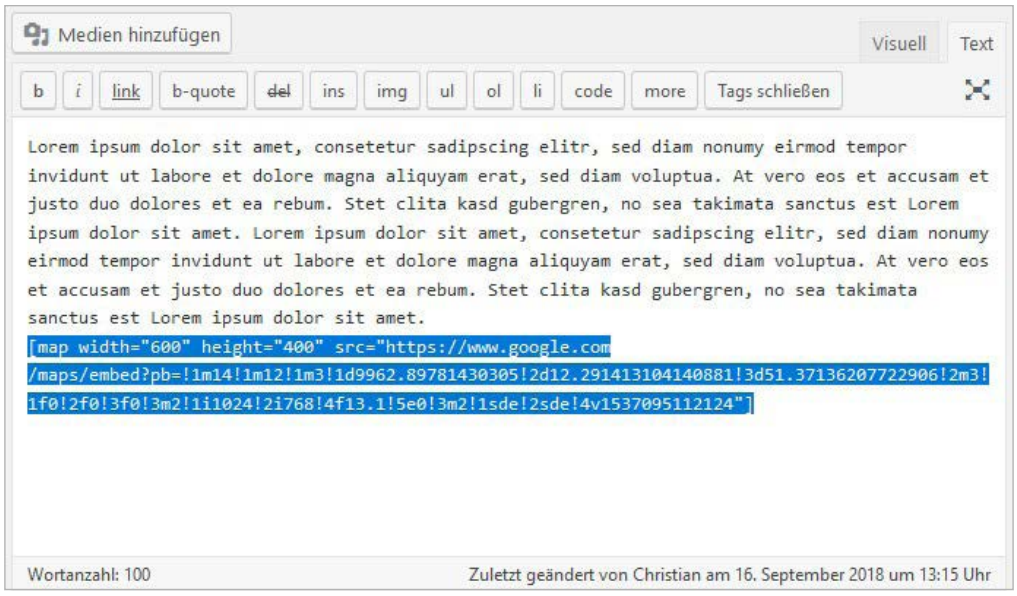
Von nun an fügst Du neue Code-Schnipsel in dieses Plugin ein, statt sie in die **functions.php** Deines Themes zu kopieren. Gegebenenfalls kannst Du natürlich bereits früher eingefügten Code aus der **functions.php** hierher umziehen.

Die Möglichkeiten sind vielfältig, von kurzen Code-Schnipseln mit nützlichen Detailfunktionen über Shortcodes bis hin zu Custom Post Types – also alles, was Du sonst auch in der **functions.php** tun kannst.

Um zu demonstrieren, wie Dein Plugin zum Leben erwacht, definieren wir beispielhaft einen neuen Shortcode. Wir möchten Google-Landkarten in Postings einbauen und deren Breite und Höhe jeweils individuell definieren. Füge dazu im Gutenberg-Editor einen Shortcode-Block ein. Der Shortcode dazu soll so aussehen:

```
[map width="600" height="400" src="https://www.google.com/maps/embed?..."]
```

An die Stelle von »https://www.google.com/maps/embed?...« gibst Du die URL ein, die Du bei Google Maps (Burger-Menü > **Karte teilen oder einbetten > Karten einbetten**) abrufst. Als Source (src) fügst Du dabei nur die Karten-URL (https) ein (ohne iframe, width, height etc.).

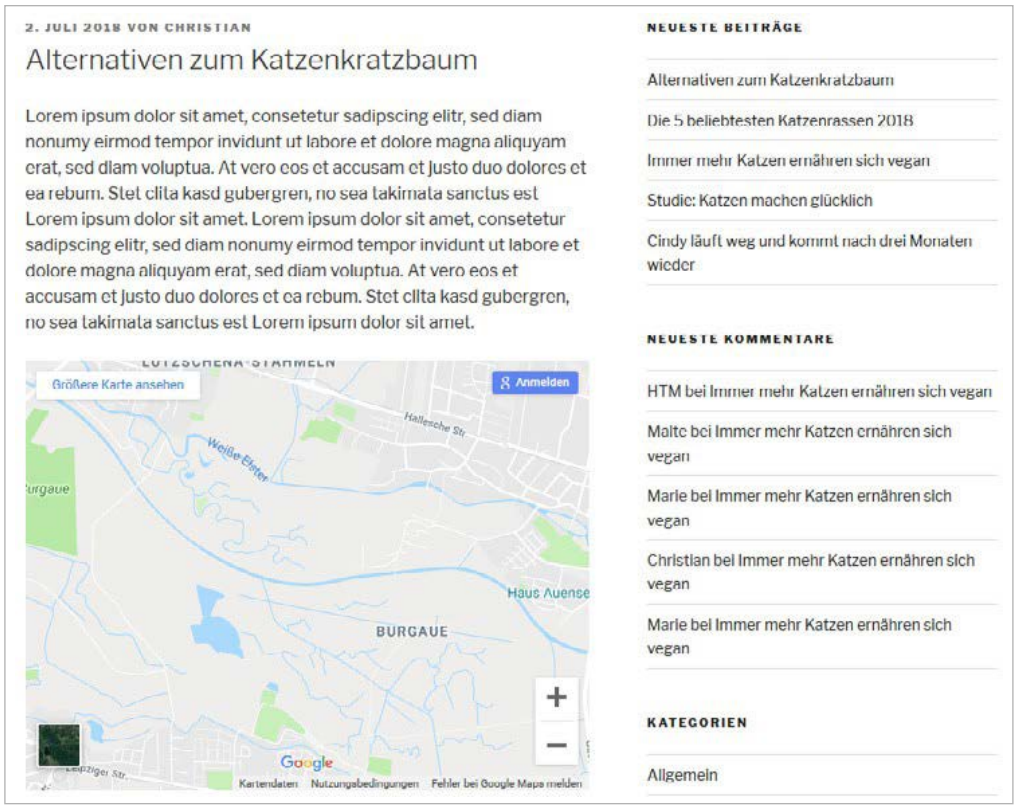


Über den Shortcode definierst Du die Breite, Höhe und Karten-URL.

Das Plugin mit dem dazugehörigen PHP-Code sieht dann so aus:

```
/* Shortcode [map width="" height="" src=""] - Google-Karte einfuegen */
function MeineGoogleMap($atts) {
      extract(shortcode_atts(array(
         „width“ => ‚800‘,
         „height“ => ‚600‘,
         „src“ => ‚‘
         ), $atts));
      return ‚<iframe width="‘.$width.‘" height="‘.$height.‘" frameborder="0“
      scrolling="no“ marginheight="0“ marginwidth="0“ src="‘.$src.‘"></iframe>‘;
}
add_shortcode(„map“, „MeineGoogleMap“);
```

Die Shortcode-Funktion in Deinem Plugin erledigt den Rest:



Welcher Code gehört ins Plugin, welcher nicht?

Gut aufgehoben ist in Deinem Plugin sämtlicher Code, der nichts direkt mit Deinem Theme zu tun hat. Das sind also Funktionen, die auch unabhängig vom Theme nützlich sind.

Im Umkehrschluss solltest Du alles, was direkten Einfluss auf das Theme nimmt, auf dessen Funktionen aufbaut oder solche verändert, in der **functions.php** belassen. Ein typisches Beispiel dafür wäre die Definition zusätzlicher Bildgrößen für die Medien-Bibliothek.

Denn: Wechselst Du später einmal das Theme, wären diese Code-Teile im Plugin heimatlos und würden WordPress mit unnötigem, weil dann nutzlosem Code aufblähen.

Tipp: Strukturiere Dein Plugin mit Kommentarzeilen

Gewöhnlich enthält ein Plugin genau eine Funktionalität. Im Falle Deines individuellen Plugins packst Du aber viele voneinander unabhängige Code-Schnipsel in ein einziges Plugin. Damit Du Dich später darin noch zurechtfindest, solltest Du Ordnung halten und alles gut dokumentieren – mit Kommentaren und auch außerhalb des Plugins, um dessen Code nicht aufzublähen.

Gib innerhalb des Plugins jeder Funktion per PHP-Kommentar **/* kommentar */** einen Namen oder eine Kurzbeschreibung und lege Dir dann separat eine Textdatei an, in die Du Details zur jeweiligen Funktionalität hinterlegst. Besonders wichtig ist dabei die Quelle, aus der Du den Code-Schnipsel bezogen hast. Denn dann kannst Du später bei eventuell auftretenden Problemen dort nachsehen und eventuelle Updates finden.

3.2 Plugins ersetzen durch Code am Beispiel »Related Posts«

Mit sogenannten »Related Posts«-Plugins kannst Du bequem weitere Beiträge mit gleichen Schlagworten anzeigen. Plugins müssen allerdings aktualisiert werden, sind potenzielle Sicherheitslücken und können Dein Blog ausbremsen. Es geht aber auch anders – ohne Plugin und Nebenwirkungen.

Um nach dieser Anleitung Beiträge unterhalb eines Beitrags mit Bild anzuzeigen, musst Du diese mit einem Beitragsbild und (identischen) Schlagworten versehen.

Ähnliche Beiträge anzeigen

Zunächst legst Du fest, wo die ähnlichen Beiträge angezeigt werden sollen. Dafür bietet sich etwa der Bereich zwischen Beitrag und Kommentarbereich an. Wo der Code genau eingefügt wird, hängt vom verwendeten Theme ab. Bei Twenty Twenty kannst Du die Zeilen in die **/template-parts/comments.php** ganz oben direkt nach dem Kommentar

```
/*
 * If the current post is protected by a password and
 * the visitor has not yet entered the password we will
 * return early without loading the comments.
*/
```

einfügen.

```
// Ähnliche Beiträge anzeigen
$orig_post = $post; global $post;
$tags = wp_get_post_tags($post->ID); if ($tags) {
$tag_ids = array();
foreach($tags as $individual_tag) $tag_ids[] = $individual_tag->term_id;
$args=array(
‚tag__in‘ => $tag_ids,
‚post__not_in‘ => array($post->ID),
‚date_query‘ => array( array( ‚after‘ => ‚-1 year‘ ) ), // Nur Beiträge jünger als ein Jahr
‚posts_per_page‘=>3, // Anzahl der angezeigten Beiträge
‚caller_get_posts‘=>1,
```

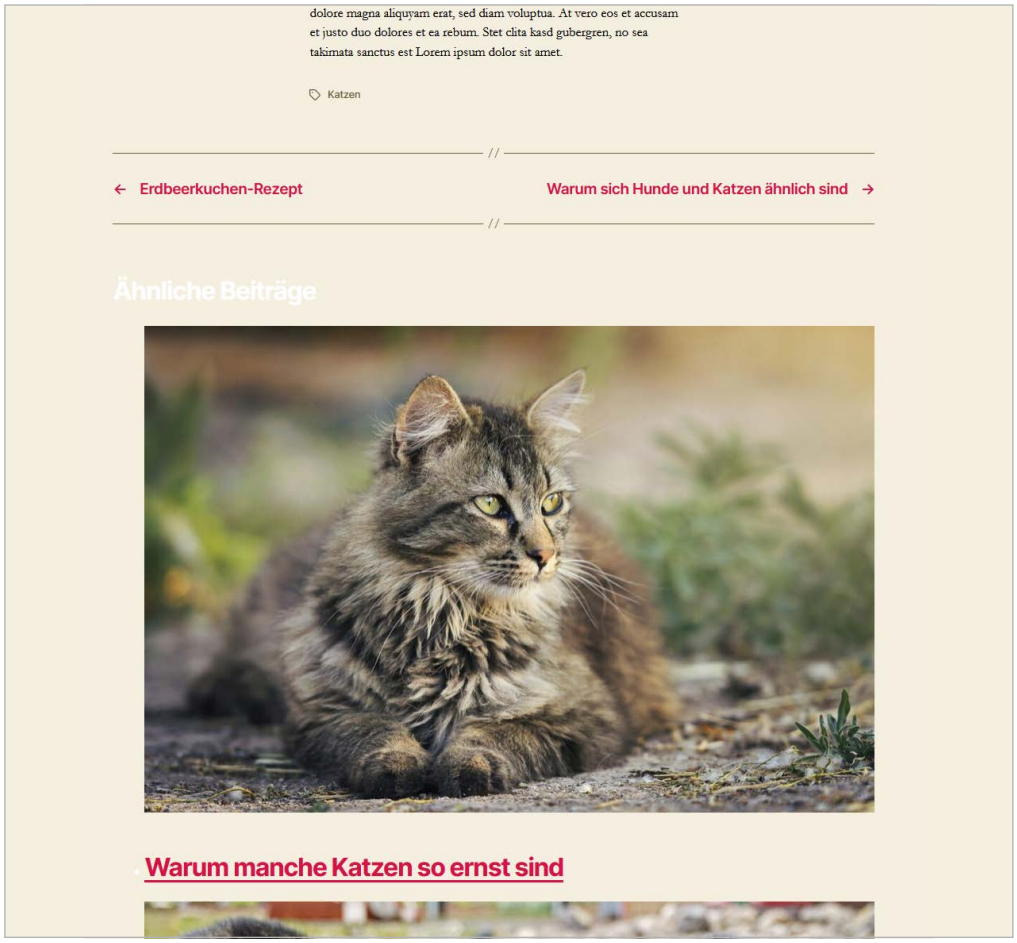
```
,orderby'=>'rand'
);
$my_query = new wp_query( $args ); if( $my_query->have_posts() ) {

echo ,<div id="related-posts"><h3><b>Ähnliche Beiträge</b></h3><ul>';

while( $my_query->have_posts() ) {
$my_query->the_post(); ?>

<li><div class="related-images"><a href="<? the_permalink()?>" rel="bookmark"
title="<?php the_title(); ?>"><?php the_post_thumbnail(); ?></a></div>
<div class="related-content">
<h3><a href="<? the_permalink()?>" rel="bookmark" title="<?php the_title();
?>"><?php the_title(); ?></a></h3>
</div>
</li>
<? }
echo ,</ul></div>';
}
}
$post = $orig_post; wp_reset_query();
```

Bei Twenty Twenty sieht das im Editor so aus:

```
/*
 * If the current post is protected by a password and
 * the visitor has not yet entered the password we will
 * return early without loading the comments.
*/

//Ähnliche Beiträge anzeigen
$orig_post = $post; global $post;
$tags = wp_get_post_tags($post->ID); if ($tags) {
$tag_ids = array();
foreach($tags as $individual_tag) $tag_ids[] = $individual_tag->term_id;
$args=array(
'tag__in' => $tag_ids,
'post__not_in' => array($post->ID),
'date_query' => array( array( 'after' => '-1 year' ) ), //Nur Beiträge jünger als ein Jahr
'posts_per_page'=>3, //Anzahl der angezeigten Beiträge
'caller_get_posts'=>1,
'orderby'=>'rand'
);
$my_query = new wp_query( $args ); if( $my_query->have_posts() ) {

echo '<div id="related-posts"><h3><b>Ähnliche Beiträge</b></h3><ul>';

while( $my_query->have_posts() ) {
$my_query->the_post(); ?>

<li><div class="related-images"><a href="<? the_permalink()?>" rel="bookmark" title="<?php the_title(); ?>"><?php the_post_thumbnail(); ?></a></div>
<div class="related-content">
<h3><a href="<? the_permalink()?>" rel="bookmark" title="<?php the_title();
?>"><?php the_title(); ?></a></h3>
</div>
</li>
<? }
echo '</ul></div>';
}
}
$post = $orig_post; wp_reset_query();


if ( post_password_required() ) {
    return;
}

if ( $comments ) {
    ?>
```

Die mit »//Nur Beiträge jünger als ein Jahr« und »//Anzahl der angezeigten Beiträge« kommentierten Zeilen sind selbsterklärend. Auf diese Weise werden drei ähnliche Beiträge angezeigt, die nicht älter als ein Jahr sind. Um Leser nicht abzulenken, sind weniger Beiträge besser als mehr.

Unterhalb der Artikel werden nun Beiträge in zufälliger Reihenfolge angezeigt, die mindestens ein identisches Schlagwort haben. In diesem Beispiel sind das Beiträge mit dem Schlagwort »Katzen«. Das Beitragsbild und der Titel werden automatisch verlinkt:



Darstellung anpassen

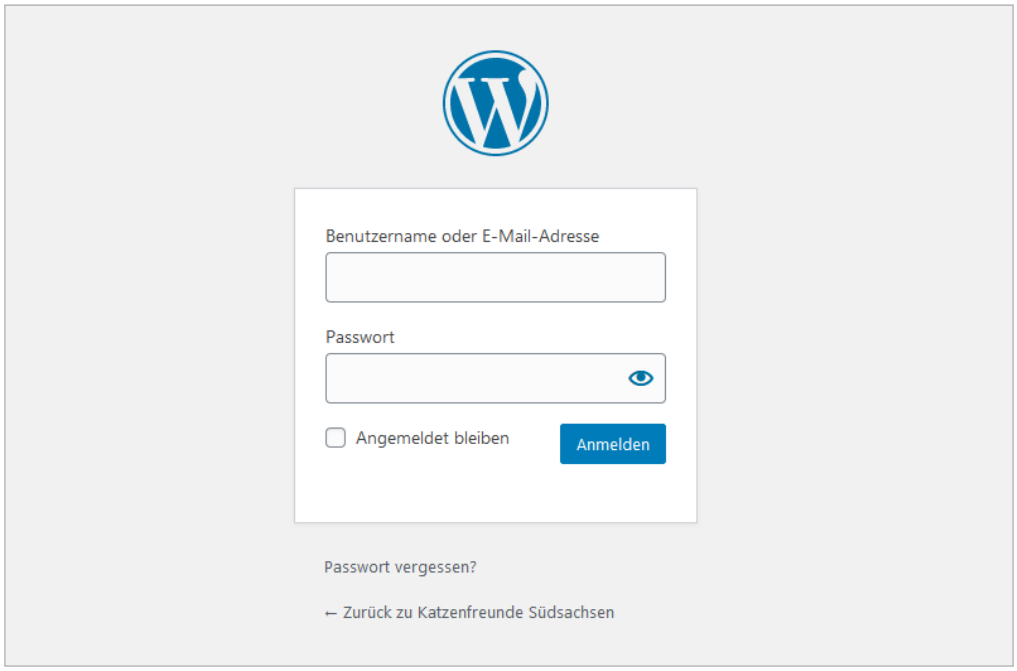
Die Bilder fügen sich zwar in das bestehende Layout ein. Noch sind die Bilder aber zu groß, die Abstände passen noch nicht und die Listenpunkte links neben den Bildern stören. Mit ein wenig Feinjustierung per CSS ist auch das kein Problem. Den Code fügst Du einfach in die **style.css** Deines Child-Theme ein:

```
/* Ähnliche Beiträge ausrichten*/
#related-posts {
position: relative;
margin-right: auto;
margin-left: auto;
max-width: 58rem;
}
/* Ähnliche Beiträge Listenpunkte entfernen*/
#related-posts li {
list-style: none !important;
margin-left: -3rem;
}

/* Ähnliche Beiträge Abstände */
.related-content h3 {
   margin: 1rem auto 6rem;
}
/* Farbe „Ähnliche Beiträge“ */
#related-posts h3 {
   color: #000000;
}
```

Deine Anpassungen solltest Du auf dem Desktop und dem Smartphone bzw. mit den **›Entwickler-Werkzeugen Deines Browsers** testen. Mobil sehen die ähnlichen Beiträge bei uns damit so aus:

4 Backend

4.1 WordPress-Login bunter machen

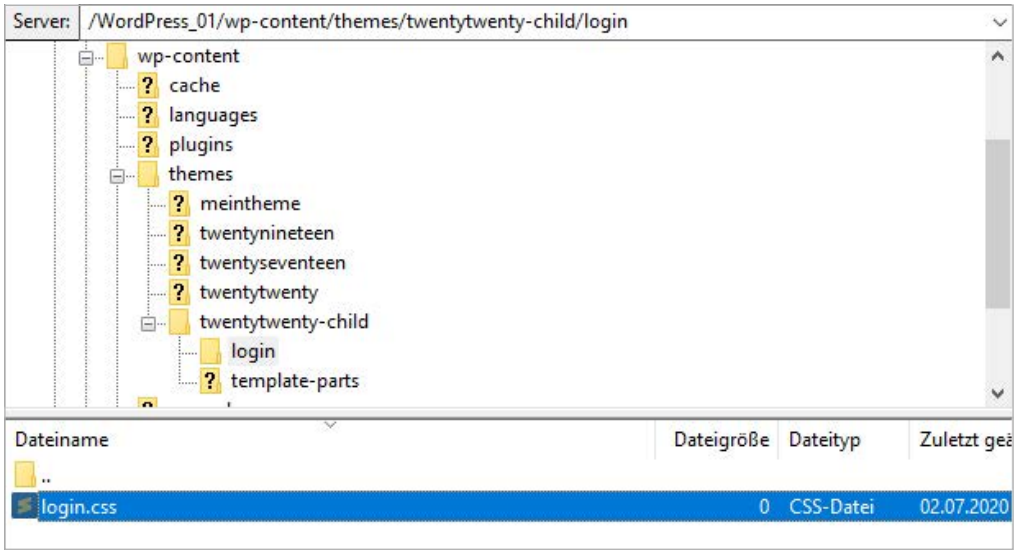


Standard-Login von WordPress

Grauer Hintergrund, WordPress-Logo und schlichtes Formular müssen nicht sein. Schließlich kannst Du neben **›Theme** und **›Backend** auch den Login-Bereich von WordPress an Deine Bedürfnisse anpassen. Du hast beispielsweise die Möglichkeit, ein Hintergrundbild einzufügen, ein eigenes Logo zu nutzen und das Formular anzupassen.

Wie kann ich den Login-Bereich anpassen?

Für die Anpassungen benötigst Du zunächst eine CSS-Datei, die Du am besten in einem Unterordner Deines Child-Themes ablegst. In unserem Beispiel erstellen wir dazu das Verzeichnis »login« und speichern darin die Datei **login.css**. Der Pfad sieht beim Child-Theme von Twenty Twenty dann so aus: /wp-content/themes/twentytwenty-child/login/login.css.



Die Datei login.css wird im Child-Theme abgelegt.

Damit das Theme die nachfolgenden Änderungen übernimmt, gibst Du anschließend folgende Zeilen ans Ende der **functions.php** Deines Child-Themes ein:

```
//Layout Login-Bereich
function strato_login() {
echo ‚<link rel="stylesheet" type="text/css" href="' . get_bloginfo(‚stylesheet_
directory') . ‚/login/login.css" />';
}
add_action(‚login_head', ‚strato_login');
```

Hintergrundbild einfügen und Farben ändern

Nachdem das Child-Theme vorbereitet ist, kannst Du in die **login.css** beliebige Anweisungen eingeben. Falls Du ein Hintergrundbild verwenden möchtest, lädst Du das am besten in den Ordner »login«. Mit diesem Code fügen wir zum Beispiel ein Hintergrundbild ein und färben die Links unterhalb des Formulars weiß:

```
body.login {
   background: url(,/wp-content/themes/twentytwenty-child/login/login-
   hintergrund.jpg‘);
   background-repeat: no-repeat;
   background-attachment: fixed;
   background-position: center;
}

.login #backtoblog a, .login #nav a {
   color: #ffffff;
}
```

WP Logo durch Katzenbild ersetzen

Das sieht schon besser aus, es fehlt aber noch das Katzenbild (oder Logo). Dazu lädst Du eine 84x84 Pixel große Datei namens **login-katze.png** in den login-Ordner. Mit folgenden CSS-Zeilen bindest Du das Katzenbild ein:

```
#login h1 a {
background-image: url(,/wp-content/themes/twentytwenty-child/login/login-
katze.png‘);
}
```

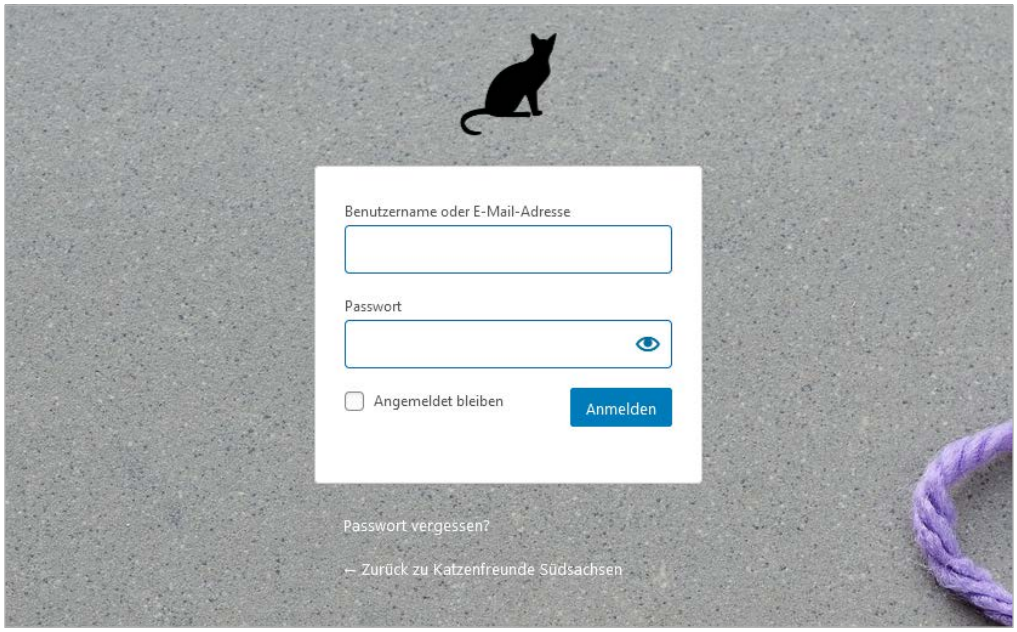
Login-Bereich mit WordPress-Logo (links) und Katzenbild

Eingabefelder des Formulars abrunden

Darüber hinaus kannst Du auch den Login-Bereich weiter anpassen: Du hättest das Formular und die Eingabefelder gern abgerundet statt eckig? Die Felder mit blauen statt grauen Rahmen? Kein Problem mit diesem Code:

```
.login label {
    font-size: 12px;
    color: #555555;
}

.login input[type="text"]{
    background-color: #ffffff;
    border-color:#0164A5;
    -webkit-border-radius: 4px;
}

.login input[type="password"]{
    background-color: #ffffff;
    border-color:#0164A5;
    -webkit-border-radius: 4px;
}

.login form {
    -webkit-border-radius: 4px;
}
```

Das Ergebnis ist ein WordPress-Login mit Hintergrundbild, Katze und einem schöneren Formular:



Login kann auch bunt sein

Es ist so einfach, den Standard-Login von WordPress aufzuwerten. Mit wenigen Handgriffen schaffst Du eine Login-Seite, die sich im wahrsten Sinne des Wortes von der grauen Masse abhebt. Zusammen mit den Backend-Anpassungen fühlt sich WordPress damit viel mehr wie etwas Eigenes, Individuelles an.

4.2 Backend anpassen: Individuell statt von der Stange

Keine Lust mehr auf das Standard-Grau? Neben dem Theme (Frontend) lässt sich auch das Backend von WordPress flexibel individualisieren. So kannst Du beispielsweise Farben und Menüpunkte ändern und ein eigenes Logo einfügen. Das wirkt professioneller und beschleunigt Deinen Workflow.

Bitte vorher ein **›Backup** aller WordPress-Dateien machen, damit die Website im Notfall wiederhergestellt werden kann! Die genannten PHP-Code-Beispiele fügst Du einfach ans Ende der functions.php des **›Child-Themes** ein.

Farben ändern

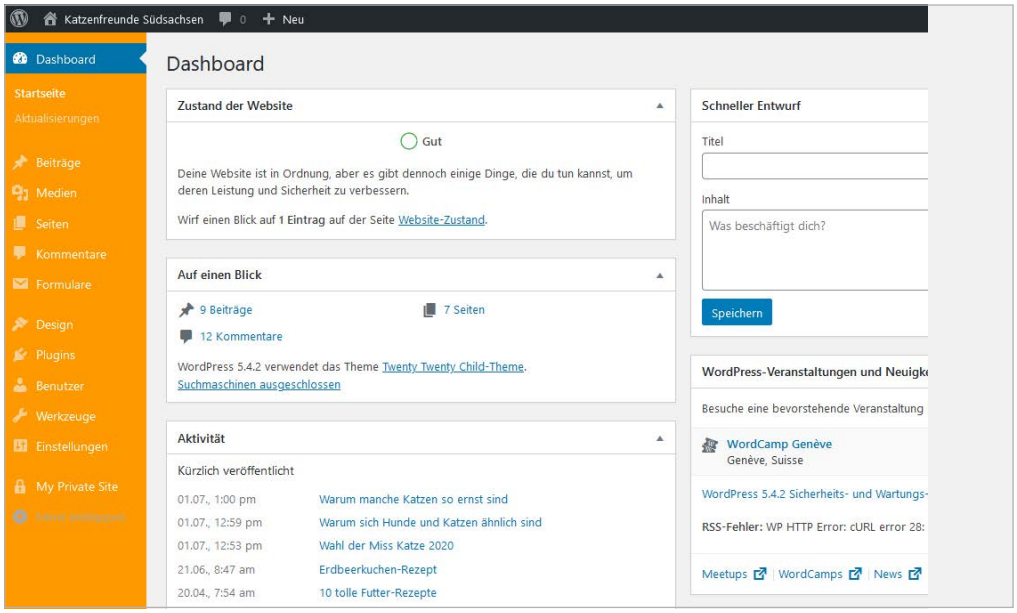
Das grau-schwarz-blaue Standard-Farbschema des WordPress-Backends ist funktional, aber auch Geschmackssache. Über **Benutzer > Dein Profil** stehen zwar alternative Vorlagen zur Auswahl, mit einer eigenen CSS-Datei für das Backend bist Du aber deutlich flexibler. Diese wird mit folgenden Zeilen in das Theme eingebunden:

```
// admin.css einbinden
function admin_style() {
wp_enqueue_style(‚admin-styles‘, get_stylesheet_directory_uri().‘/admin.css‘);
}
add_action(‚admin_enqueue_scripts‘, ‚admin_style‘);
```

In diesem Beispiel nennen wir die CSS-Datei **admin.css** und legen sie im Hauptverzeichnis des Child-Themes ab. Mithilfe des folgenden CSS-Codes färbst Du das Admin-Menü auf der rechten Seite und die Untermenüs orange:

```
#adminmenu, #adminmenu .wp-submenu, #adminmenuback, #adminmenuwrap {
    background-color: #ff9900 !important;
}
#adminmenu div.wp-menu-name {
    color: #ffffff;
}
```

So sieht das Ergebnis aus:



Menü ausblenden

Das Menü kannst Du ebenfalls anpassen. Benötigst Du bestimmte Punkte nicht, kannst Du sie nach dem Muster **remove_menu_page(‚XXX.php‘);** einfach ausblenden. Bitte beachte, dass das Menü dabei aktiv bleibt und die Funktionen weiterhin über die URL aufrufbar sind. Um Zugriffsrechte für bestimmte Benutzer zu beschränken, solltest Du die entsprechenden Rollen von WordPress nutzen (also statt Administrator z. B. Autor oder Mitarbeiter).

Die nächsten Code-Zeilen dienen der Übersicht, welche Funktionen Du wie ausblendest. Anhand der Kommentierung siehst Du, welche Zeile für welchen Bereich zuständig ist. Würdest Du den kompletten Code übernehmen, wäre Deine Menüleiste leer.

```
function strato_remove_menus(){
   remove_menu_page( ‚index.php‘ );                  //Dashboard
   remove_menu_page( ‚edit.php‘ );                   //Beiträge
   remove_menu_page( ‚upload.php‘ );                 //Medien
   remove_menu_page( ‚edit.php?post_type=page‘ );    //Seiten
   remove_menu_page( ‚edit-comments.php‘ );          //Kommentare
   remove_menu_page( ‚themes.php‘ );                 //Design
   remove_menu_page( ‚plugins.php‘ );                //Plugins
   remove_menu_page( ‚users.php‘ );                  //Benutzer
   remove_menu_page( ‚tools.php‘ );                  //Werkzeuge
   remove_menu_page( ‚options-general.php‘ );        //Einstellungen
}
add_action( ‚admin_menu‘, ‚strato_remove_menus‘ );
```

Möchtest Du beispielsweise **Plugins** und **Einstellungen** ausblenden, gibst Du folgenden Code ein:

```
function strato_remove_menus(){
   remove_menu_page( ‚plugins.php‘ );                //Plugins
   remove_menu_page( ‚options-general.php‘ );        //Einstellungen
}
add_action( ‚admin_menu‘, ‚strato_remove_menus‘ );
```

Damit die Änderungen sichtbar werden, musst Du ggf. den Cache löschen oder Dich neu einloggen.

Menüs umbenennen

Du kannst die Menüpunkte nicht nur ausblenden, sondern auch umbenennen. Um etwa »Beiträge« durch einen anderen Begriff zu ersetzen, gehst Du folgendermaßen vor:

```
function strato_post_menu_rename() { global $menu;
$menu[5][0] = ‚News‘;
}
add_action( ‚admin_menu‘, ‚strato_post_menu_rename‘ );
```

In diesem Fall nennen wir das Menü »Beiträge« in »News« um.

WordPress ordnet jedem Menü einen Key zu, der hier angegeben werden muss. Das Menü »Beiträge« hat den Key 5, danach folgt Key 0 für die **index.php** (siehe Zeile 3). Auf diese Weise kannst Du sämtliche Menüpunkte im WordPress-Backend umbenennen. Eine Übersicht über die benötigten Keys findest Du **›hier**.

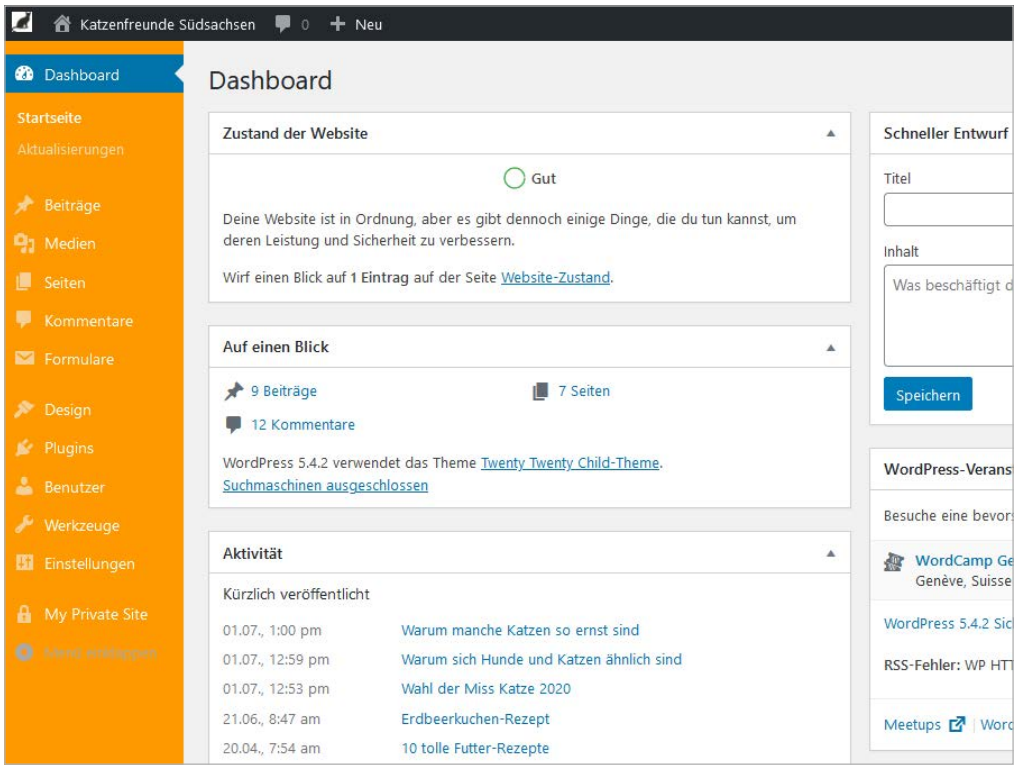
Katzen-Logo einfügen

Es gibt verschiedene Wege, ein Logo im Backend anzuzeigen. Der einfachste besteht darin, das kleine WordPress-Logo in der Ecke oben links auszutauschen. Dazu legst Du eine 18×18 Pixel große Datei als **backend-katze.png** im Hauptverzeichnis Deines Child-Themes ab und ergänzt die **functions.php** um folgende Zeilen:

```
// Backend Logo
function wpb_custom_logo() { echo ,
<style type=“text/css“>
#wpadminbar #wp-admin-bar-wp-logo > .ab-item .ab-icon:before {
background-image: url(, . get_bloginfo(‚stylesheet_directory‘) . ‚/backend-katze.
png) !important;
background-position: 0 0; background-repeat: no-repeat; color:rgba(0, 0, 0, 0);
}
#wpadminbar #wp-admin-bar-wp-logo.hover > .ab-item .ab-icon {
background-position: 0 0;
}
</style>
```

```
,;
}

add_action(‚wp_before_admin_bar_render‘, ‚wpb_custom_logo‘);
```

Danach sieht das Backend so aus:



5 Sicherheit

5.1 Strategischer Hintergrund: Welche Sicherheitslücken solltest Du schließen?

Um Dein Blog vor Hackern zu schützen, bedarf es mehr als ein paar Sicherheitsmaßnahmen in WordPress selbst. Nur mit einer Sicherheits-Strategie, die Deine sämtlichen Online-Aktivitäten umfasst, bist Du wirklich geschützt.

Wenn ein Angreifer sich auf anderem Weg beispielsweise Dein Admin-Passwort besorgen kann oder das WordPress-Login einfach umgeht, helfen alle WordPress-Sicherheitsmaßnahmen nichts. Deshalb ist es noch wichtiger, sich eine Sicherheitsstrategie zuzulegen, die auch das gesamte Umfeld absichert: ein FTP-Zugang, der Login zum STRATO-Backend, der eigene E-Mail-Account und einiges mehr.

Aber ist das nicht übertrieben? Schließlich hast Du nichts für Hacker Interessantes – denkst Du. Was Hacker jedoch tatsächlich suchen, sind Ressourcen, die sie für illegale Zwecke missbrauchen können, ohne dabei entdeckt zu werden: Sie wollen zum Beispiel illegale Dateien, Bilder und Videos tauschen oder schädlichen Code verbreiten. Um ihr Ziel zu erreichen, platzieren sie außerdem unbemerkt Spam-Links oder tauschen Werbebanner und Affiliate-Codes aus, sodass nicht mehr Du, sondern der Hacker die Vermittlungsprovisionen kassiert.

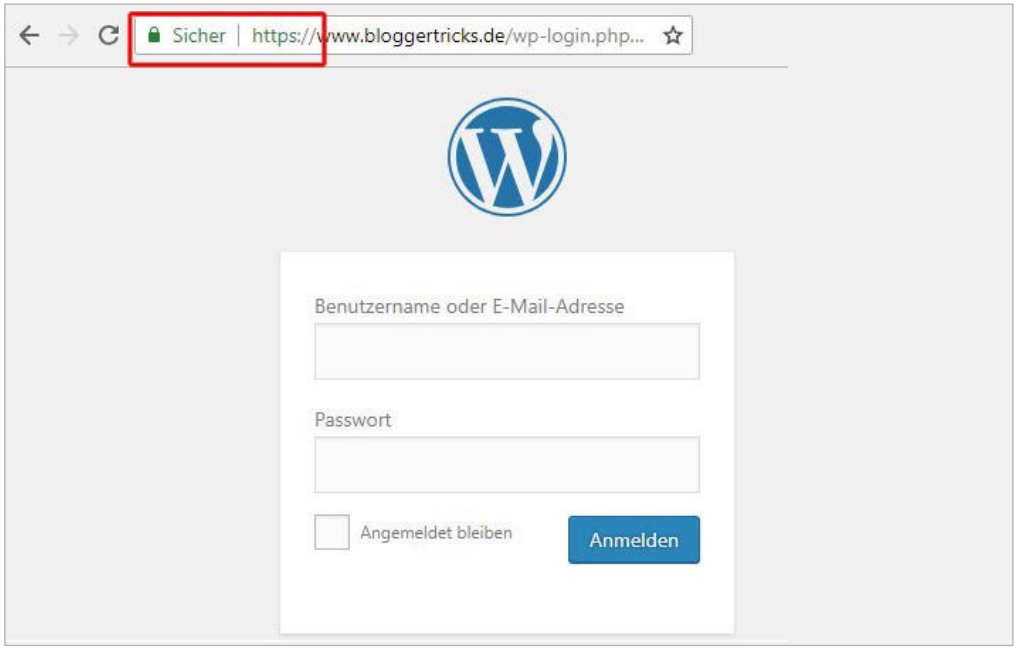
Hacker-Angriffe geschehen automatisiert und systematisch. Fällt einem Hacker beispielsweise Dein E-Mail-Passwort in die Hände, weil Dein E-Mail-Provider gehackt wurde, oder filtert er in einem offenen WLAN Dein FTP-Passwort heraus, kann er mit diesen Informationen unter Umständen kompletten Zugriff auf Deinen Webspace bekommen.

Deshalb ist es wichtig, dass Du Dich generell möglichst sicher durchs Internet bewegst und keine einfachen Angriffspunkte bietest. Einige der nachfolgend aufgezeigten Maßnahmen sind sehr einfach umzusetzen, andere bedeuten etwas Aufwand oder die Umstellung Deiner Gewohnheiten.

Verschlüsselte Verbindungen

Greife auf sicherheitsrelevante Websites wie beispielsweise WordPress, Dein Webspace per FTP, Deinen E-Mail-Account oder Dein Onlinebanking nur über sichere Verbindungen zu. Das kann am Handy eine Datenverbindung Deines Mobilfunkanbieters sein oder Dein eigener Internetzugang zu Hause.

Bist Du in fremden WLANs unterwegs, sind zusätzliche Sicherheitsmaßnahmen unabdingbar. Grundsätzlich lassen sich die Daten in einem offenen WLAN von jedem mitlesen, der ebenfalls mit diesem WLAN verbunden ist.



Der Login bei WordPress findet sinnvollerweise nur über eine SSL-Verbindung statt.

Mindestens solltest Du also Passwörter nur über gesicherte Verbindungen übertragen, also SSL-verschlüsselt. Bei den meisten E-Mail-Anbietern ist das der Fall, für Dein eigenes WordPress musst Du die **›SSL-Verschlüsselung** aber selbst einrichten.

Solltest Du dennoch einmal ein Passwort über eine unverschlüsselte Verbindung nutzen müssen, ändere das Passwort hinterher so bald wie möglich über eine sichere Verbindung.

VPN-Verbindung nutzen

Deutlich sicherer fährst Du mit einer sogenannten VPN-Verbindung. Dabei stellt Dein Laptop oder Smartphone eine verschlüsselte Verbindung zu einem Server des VPN-Dienstleisters her. Er überträgt alle Daten komplett verschlüsselt und damit für Angreifer quasi unsichtbar zunächst auf diesen Server. Erst von dort aus gehen die Daten dann an den eigentlichen Bestimmungsort, also beispielsweise zu Deinem

WordPress-Login. Ein Hacker kann so – anders als bei einer SSL-Verschlüsselung – nicht einmal sehen, was Du überhaupt im Internet machst.

VPN-Dienste gibt es in gewissem Umfang kostenlos, beispielsweise das renommierte, in der Schweiz ansässige **›Proton VPN**. VPNs mit schneller Datenübertragung und unbegrenztem Volumen kosten allerdings monatliche Gebühren ab etwa 5 Euro. Auch manche Desktop-Firewalls bringen inzwischen eigene VPN-Funktionen mit. Wer zu Hause mit einer Fritzbox surft, kann sich von unterwegs **›per VPN mit der Fritzbox verbinden** und auf diesem Weg die sichere heimische Internetverbindung nutzen.

Sichere Passwörter

Eigentlich selbstverständlich, aber aus Bequemlichkeit oft vernachlässigt: Nutze für alle Accounts **›wirklich sichere Passwörter** und verwende für jeden Zugang ein anderes Passwort. Denn nutzt Du identische Passwörter beispielsweise für die WordPress-Datenbank, Deinen STRATO Zugang, den FTP-Zugriff auf Deinen Webspace oder den E-Mail-Account, vervielfacht sich für einen Hacker die Chance, dieses eine Passwort zu knacken und damit Zugriff auf all Deine Accounts zu bekommen.

Auch wenn Dein E-Mail-Account auf den ersten Blick nichts mit WordPress zu tun hat: Über die WordPress-Funktion zum Wiederherstellen des Passwortes ist es ein Leichtes, über Deinen E-Mail-Account die Kontrolle über Deine WordPress-Installation zu übernehmen.

FTP: nur verschlüsselt

Besonders leicht übersieht man die Risiken bei der Datenübertragung per FTP. Gängige FTP-Clients wie Filezilla übertragen Daten nämlich standardmäßig mit dem unverschlüsselten FTP-Protokoll und speichern FTP-Passwörter unverschlüsselt auf der Festplatte.

In Filezilla kannst Du SFTP (SSH File Transfer Protocol) über die Verbindungs-einstellungen aktivieren. Als Server legst Du dort **ssh.strato.de** fest, als Protokoll wählst Du **SFTP**. Der Benutzername ist Deine Domain, das Passwort das STRATO Masterpasswort. Das legst Du im STRATO Backend unter **Sicherheit > Passwörter festlegen > Masterpasswort** fest.

So stellst Du SFTP in Filezilla ein.

Auch wenn es etwas Mühe macht: Erlaube Browsern und FTP-Clients niemals, Passwörter auf der Festplatte zu speichern. Gib sie stattdessen bei der jeweiligen Verbindung immer neu ein. Denn zu leicht könnte diese Passwort-Datei in falsche Hände geraten.

Veraltete Dateien am Webserver

Besonders beliebt bei Hackern sind veraltete und vergessene Dateien auf Webservern. Eventuell hast Du früher einmal irgendeine App ausprobiert oder ein zusätzliches WordPress zum Testen installiert, das Du längst nicht mehr nutzt. Auch wenn diese Dateien von keiner Webseite her verlinkt sind: Hacker kennen die Standardpfade und Dateinamen von alten App-Versionen mit Sicherheitslücken und finden solche Dateien über automatisierte Such-Scripts.

Kümmere Dich deshalb um Datenhygiene in Deinem Webspace: Lösche alles, was Du nicht mehr benutzt, und halte alles andere mit Updates ständig auf dem aktuellsten Stand.

5.2 Potenzielle Sicherheitslücken von WordPress schließen

Um WordPress sicherer zu machen, kannst Du Login-Fehlermeldungen deaktivieren, einen Verzeichnisschutz für den Login-Bereich erstellen und das Tabellenpräfix ändern.

Login-Fehlermeldungen deaktivieren

Wenn Du Dich mit falschen Benutzernamen anmeldest, gibt WordPress folgende Fehlermeldung aus: »FEHLER: Ungültiger Benutzername.« Und wenn Du den richtigen Benutzernamen und das falsche Passwort eingibst, lautet die Antwort: »FEHLER: Das Passwort, das du für den Benutzernamen XXX eingegeben hast, ist nicht korrekt.« Diese Informationen sind für Angreifer wertvoll und sollten deshalb deaktiviert werden. Um stattdessen zum Beispiel ein »Nachts sind alle Katzen grau.« einzublenden, fügst Du folgenden Code ans Ende der **functions.php** Deines Themes ein:

```
function no_wordpress_errors() { return ‚Nachts sind alle Katzen grau.‘;
}
add_filter( ‚login_errors‘, ‚no_wordpress_errors‘);
```

Verzeichnisschutz für Login-Bereich erstellen

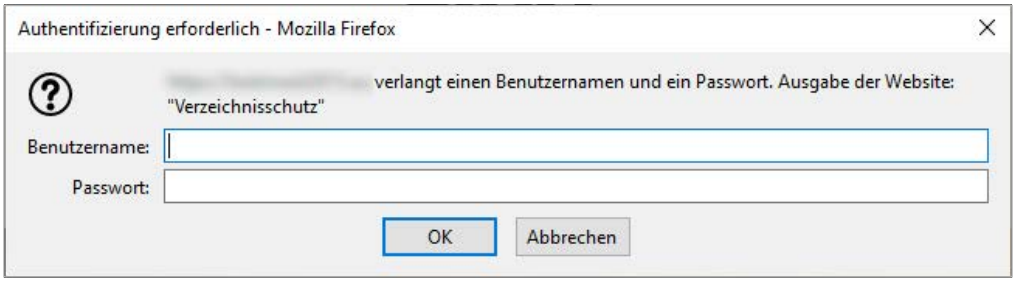
Der sensible Login-Bereich wird standardmäßig über **/wp-login.php** aufgerufen. Das wissen sich Hacker und Botnetze zunutze zu machen. Mit einem Verzeichnisschutz per **.htaccess** baust Du ein Schloss davor, das unabhängig von WordPress funktioniert. Bei einem Brute-Force-Angriff wird so die Performance Deines Blogs nicht beeinträchtigt, weil der Angreifer es gar nicht erst bis zum Login-Bereich schafft.

Dazu erstellst Du in einem Code-Editor eine leere Datei, nennst diese **.htpasswd** und öffnest den **› Htpasswd Generator**. Dort gibst Du einen beliebigen Nutzernamen und ein Passwort ein. Das Tool verschlüsselt das Passwort und gibt eine Kombination aus Nutzernamen und Datensalat aus. Diese kopierst Du in die **.htpasswd**-Datei und lädst sie in das root-Verzeichnis Deines Webspace.

Im nächsten Schritt öffnest Du die **.htaccess** und fügst ans Ende folgenden Code ein:

```
<Files wp-login.php>
AuthType Basic
AuthName „Verzeichnisschutz“
AuthUserFile /mnt/rid/25/07/56932507/htdocs/.htpasswd
Require valid-user
</Files>
```

Bei **AuthUserFile** musst Du statt ABSOLUTERPFAD Deinen individuellen absoluten Pfad zur **.htpasswd** angeben (www.wunschdomain.de/.htpasswd funktioniert nicht). Was das ist und wie Du den herausfindest, erklären wir in **›dieser FAQ**. Nach dem Upload der bearbeiteten **.htaccess** erwartet Dich beim Login-Versuch folgende Meldung:



Der Verzeichnisschutz funktioniert unabhängig von WordPress.

Damit ist Dein Login-Bereich zusätzlich geschützt. Um Dich einloggen zu können, musst Du den oben erstellten Benutzernamen und das (unverschlüsselte) Passwort eingeben. Falls Du die Zugangsdaten vergessen solltest, kannst Du die **.htpasswd**-Datei einfach durch eine neue ersetzen.

Tabellen-Präfix ändern

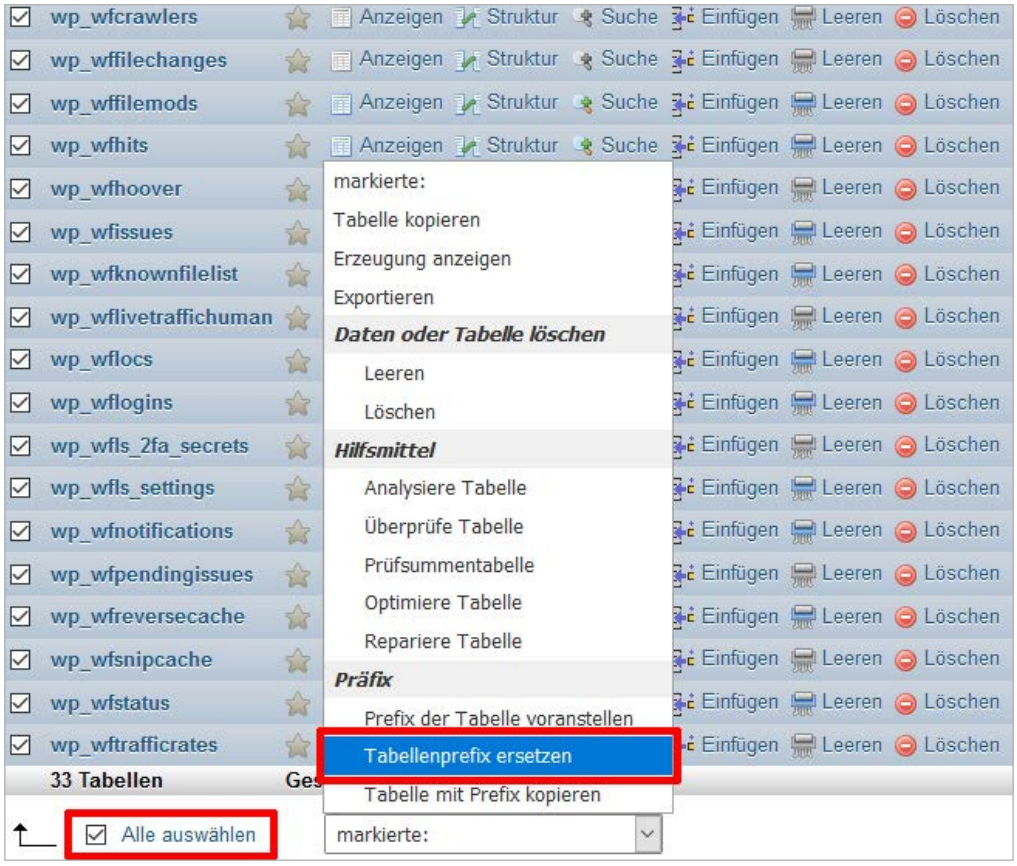
WordPress vergibt bei der Installation für die Datenbank-Tabellen standardmäßig das Präfix **wp_**. Diese potenzielle Sicherheitslücke können Dritte in Form von SQL-Injections ausnutzen. Bei einer manuellen Installation sollte deshalb am besten gleich ein anderes Präfix zum Einsatz kommen (**›siehe Kapitel 1.1**).

Vorher unbedingt ein **›Backup der Datenbank** machen, da bei einem Fehler die Website nicht mehr erreichbar ist!

So änderst Du das Tabellen-Präfix einer bestehenden WordPress-Installation:

1. Als Erstes benennst Du wie beschrieben das Präfix in der **wp-config.php** um. In unserem Beispiel wählen wir als Präfix **vogon_**, sodass die Zeile wie folgt aussieht: **$table_prefix = ‚vogon_‘;**

2. Danach rufst Du über den STRATO Kunden-Login phpMyAdmin auf: **Dein Paket > Datenbanken und Webspace > Datenbankverwaltung > verwalten** (rechts neben der entsprechenden Datenbank **> Struktur**. Klicke unten auf **Alle auswählen** und wähle daneben im Dropdown-Menü den Eintrag **Tabellenprefix** ersetzen. Hier kannst Du das alte gegen (**wp_**) das neue Präfix (hier: **vogon_**) ersetzen.



Datenbankverwaltung phpMyAdmin

3. Im Reiter SQL gibst Du anschließend die folgenden beiden Befehle ein (statt **vogon_** Dein neues Präfix):

```
UPDATE vogon_options SET option_name = REPLACE(option_name, ‚wp_‘, ‚vogon_‘);
UPDATE vogon_usermeta SET meta_key = REPLACE(meta_key, ‚wp_‘, ‚vogon_‘);
```

Welche Quellen geben Infos über Sicherheitslecks?

Welche Sicherheitslecks bei WordPress, den Plugins und Themes aktuell existieren, findest Du über verschiedene Quellen schnell heraus:

Eine gute deutsche Quelle ist über Twitter unter dem Namen **›@WPSicherheit** erreichbar. Hier gibt es Tweets über Lücken von WordPress selbst oder von Plugins mit Angabe der betroffenen Version. Taucht eines Deiner Plugins oder die WordPress-Version auf, die Du in Deinem Blog einsetzt, kannst Du so schnell reagieren und ggf. die betroffene Software aktualisieren. Existiert noch kein Update, kannst Du beim Autor nachhaken, wann es ein Sicherheitsupdate geben wird.

›ExploitDB (Englisch) ist eine weitere Quelle. Dies ist eine weltweite Datenbank für Sicherheitslücken verschiedener Programme, unter anderem auch WordPress und dessen Plugins und Themes. Gib dazu rechts oben im Suchfeld „wordpress“ ein und Du erhältst eine Liste mit bekannten Exploits – mit Angabe des Datums und allen weiteren Informationen zu den einzelnen Sicherheitslücken. In der ExploitDB kannst Du auch gezielt nach Deinen Plugin- und Theme-Versionen suchen. So findest Du heraus, ob es bekannte Sicherheitslücken gibt.

Missverständnis: »Viel hilft viel«

Bei allen Risiken solltest Du dennoch nicht dem Missverständnis aufsitzen, dass viel automatisch auch viel bringt. Absolute Sicherheit gibt es nicht und jede Sicherheits-maßnahme hat ihren Preis – ob Zeit und Nerven, Geld oder Performance- Einbußen durch Sicherheitsmaßnamen. Zwei oder drei parallel installierte Sicherheits-Plugins beispielsweise machen Dein Blog unverhältnismäßig langsam und kosten damit Besucher und Google-Ranking.

Sicherheit ist immer eine Risikoabwägung. Der Aufwand muss verhältnismäßig bleiben. Für einen Webshop mit hohen Umsätzen kann schon ein Ausfall von einer Stunde ein Desaster sein, während ein Hobby-Blogger hackerbedingt auch mal eine Woche Offline-Zeit verschmerzen kann.

Wäge also bei Sicherheitsmaßnahmen das Risiko in Deiner persönlichen Situation ab: Was passiert konkret, wenn ich gehackt werde? Und wie hoch ist der Wiederherstellungsaufwand? Für ein Hobby-Blog reichen vielleicht schon regelmäßige Backups aus, sodass sich das Blog nach einem Hackerangriff wiederherstellen lässt. Eine Shop-Website, die sensible Kundendaten verarbeitet, muss unbedingt verhindern, dass diese Daten in falsche Hände geraten.

Zu viel vermeintliche Sicherheit kann auch schaden

Viele Sicherheits-Tutorials für WordPress, die man im Internet findet, berücksichtigen diese Risikoabwägung nicht. Die dort beschriebenen, umfangreichen Maßnahmen verursachen in den meisten Fällen mehr Probleme, als sie potenziell der Sicherheit zuträglich sind.

Da wird beispielsweise empfohlen, das Plugin-Verzeichnis umzubenennen oder die Datei **wp-config.php** am Webserver in ein höheres, aus dem Web nicht zugängliches Verzeichnis zu verlegen. Weiß man hier nicht ganz genau, was man tut, und versteht die technischen Zusammenhänge nicht hundertprozentig, ist das Risiko größer als der Nutzen. Beispielsweise funktionieren manche Plugins plötzlich nicht mehr oder WordPress lässt sich überhaupt nicht mehr aufrufen.

Bei allem Streben nach Sicherheit: Das primäre Ziel ist der Spaß am Bloggen, nicht die Abwehr von Hackern. Gesundes Augenmaß ist daher wichtiger.

5.3 Typische WordPress-Fehler beheben

WordPress ist eine ausgereifte Software, die bei richtiger Anwendung kaum Probleme macht. Fehler durch Updates, Plugins, Themes oder Nutzer können dennoch dazu führen, dass WordPress eingeschränkt oder gar nicht mehr funktioniert. Wir zeigen Dir, wie Du die häufigsten Bugs behebst.

White Screen of Death

Der sogenannte White Screen of Death ist ein Fehler, mit dem viele WordPress-Blogger irgendwann konfrontiert werden. Wie der Name vermuten lässt, bleiben dabei das Front- und/oder das Backend beim Aufruf weiß. Das heißt, dass Du

WordPress nicht mehr nutzen kannst und auch Besucher nichts zu sehen bekommen. Verursacht wird das Problem häufig durch einen PHP-Fehler in einem Theme oder Plugin.

Ursache finden: Debug-Modus aktivieren

Um Informationen über die Ursache zu erhalten, solltest Du als Erstes den Debug-Modus aktivieren. Ändere dazu in der **wp-config.php** die Zeile

```
define(‚WP_DEBUG‘, false);
```

in

```
define(‚WP_DEBUG‘, true);
```

Wenn Du die Website nun erneut aufrufst, wird Dir eine detaillierte Meldung über den Fehler, die betroffene Datei und die Codezeile angezeigt.

In diesem Beispiel handelt es sich um einen Syntax-Error in der functions.php des Child-Themes.

Lösung: Datei oder Verzeichnis umbenennen

Falls Du Zugriff auf das Backend hast, solltest Du das betroffene Theme oder Plugin deaktivieren. Andernfalls kannst Du die in der Fehlermeldung benannte Datei im Webspace umbenennen (z. B. mit dem Anhang .bak), damit WordPress die fehlerhafte Datei nicht mehr aufruft. Deine Einstellungen bleiben dabei aber erhalten.

Wenn Dein Theme betroffen ist, solltest Du das gesamte Theme-Verzeichnis umbenennen. WordPress greift dann automatisch auf ein anderes Theme zurück.

Nicht nur aus diesem Grund muss immer mindestens ein weiteres Theme vorhanden sein.

Vergiss nicht den Debug-Modus anschließend wieder auszuschalten, indem Du den Wert in der oben genannten Zeile wieder zurück auf **false** setzt. Deine Besucher bekommen sonst unschöne (Fehler-)Meldungen zu sehen, die für Entwickler gedacht sind.

Tipp: Source-Code-Editoren können Dich bei der Fehleranalyse unterstützen oder sorgen dafür, fehlerhaften Code bereits bei der Eingabe aufzuzeigen. Wie Du gute Editoren findest, liest Du **›hier**.

»Dem Theme fehlt das Stylesheet style.css«

Du willst ein neues Theme installieren und bekommst die Meldung, dass das Stylesheet fehlt? Mit dem folgenden Vorgehen funktioniert die Installation:

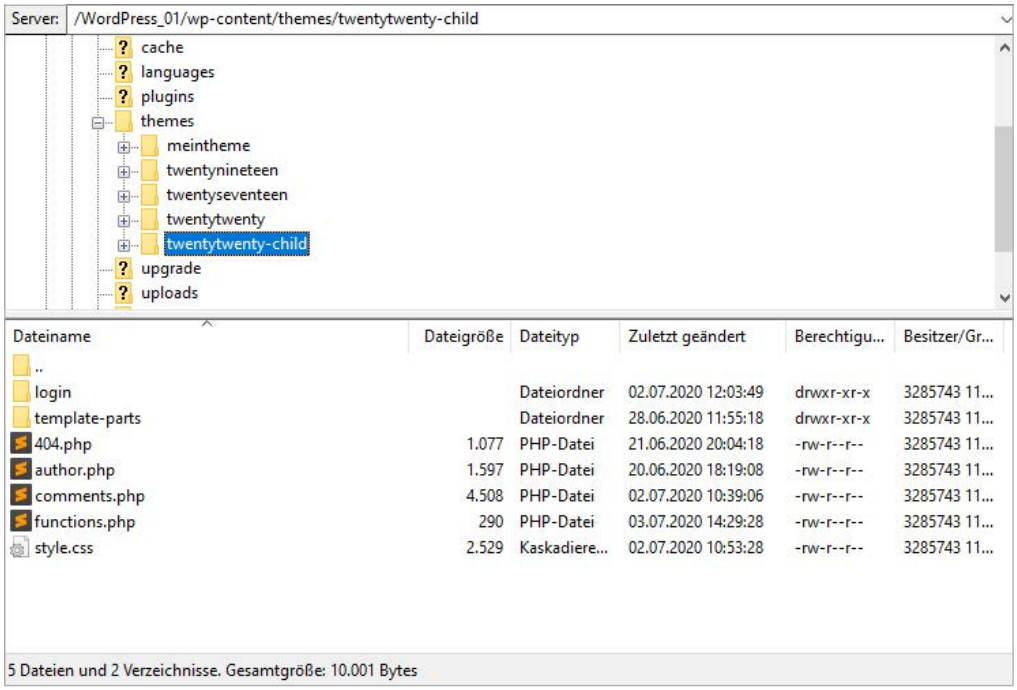
Die Fehlermeldung ist selbsterklärend: »Das Paket konnte nicht installiert werden. Dem Theme fehlt das Stylesheet style.css.« Folglich kannst Du das neue Theme nicht nutzen.

Ursache: Fehlende oder fehlerhafte style.css

Das heißt, dass WordPress die Datei **style.css** nicht finden kann oder dass die Datei fehlerhaft ist. Bei einem Theme aus der **›WordPress Repository** ist beides unwahrscheinlich. Vielleicht hast Du das (Child-)Theme ja **›selbst entwickelt**?

Lösung: Korrekte style.css bereitstellen

Stelle sicher, dass die **style.css** im Hauptverzeichnis Deines (Child-)Themes liegt. Die **style.css** und die **functions.php** sollten in der obersten Verzeichnisebene liegen.



WordPress speichert die Themes im gleichnamigen Unterverzeichnis. Wie in diesem Beispiel zu sehen ist, liegen functions.php und style.css im Hauptverzeichnis des Child-Themes.

Übrigens: Wie die **style.css** aussehen muss, haben wir **›hier beschrieben**. Das Wichtigste ist die Zeile Template: Hier muss der exakte Name des Verzeichnisses angegeben sein, in dem das ausgewählte Theme liegt. Für Twenty Twenty ist das **twentytwenty**.

404-Fehler

Sogenannte 404-Fehlerseiten treten häufig auf. Das Problem ist vergleichsweise harmlos und in WordPress schnell gefixt.

Ursache: Seite existiert nicht

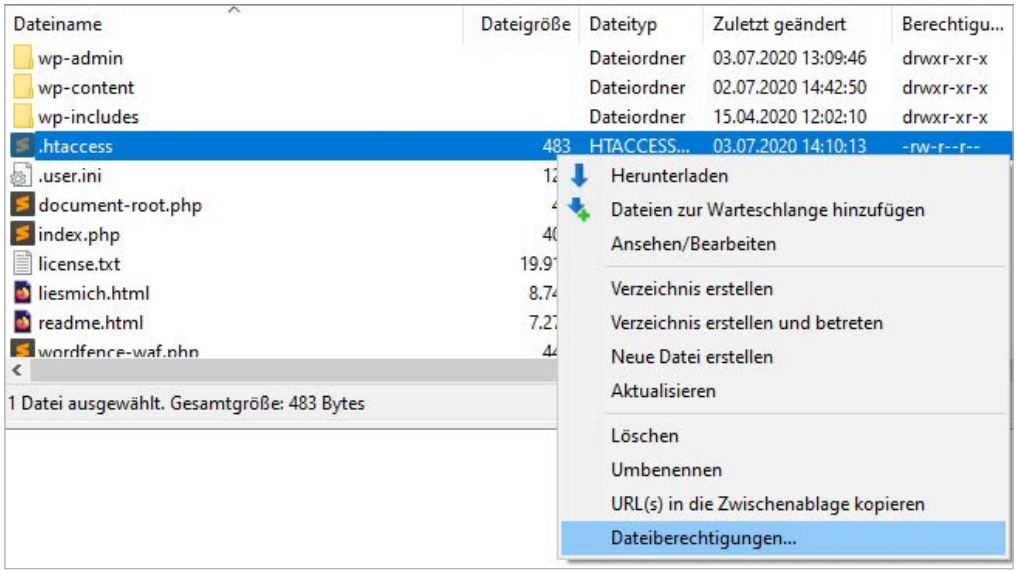
Ein 404-Error tritt auf, wenn die URL einer Seite oder eines Beitrags fehlerhaft ist oder die entsprechende Datei fehlt. Bei den meisten WordPress-Themes erscheint dann der Standard-Hinweis: »Die Seite konnte leider nicht gefunden werden.«

Mit ein paar PHP-Kenntnissen kannst Du diese Seiten sogar **›individuell gestalten**. Dennoch ist es aber natürlich besser, wenn Deine Besucher diese Seiten gar nicht erst zu sehen bekommen.

Lösung 1: ULRs neu generieren

Klicke im Backend unter **Einstellungen > Permalinks** auf **Änderungen speichern**, damit WordPress sämtliche URLs Deiner Beiträge und Seiten neu generiert. In den meisten Fällen ist das Problem damit schon behoben.

Wenn das nicht hilft, sind vielleicht die Berechtigungen der **.htaccess** im Hauptverzeichnis Deiner WordPress-Installation zu restriktiv. Klicke im FTP-Programm mit der rechten Maustaste auf die Datei und wähle dann im Menü **Dateiberechtigungen**.



Gib nun im Fenster »Dateiattribute ändern« als numerischen Wert 666 ein und wiederhole das Vorgehen oben (**Einstellungen > Permalinks > Änderungen speichern**).

Setze die Dateiberechtigungen anschließend auf die ursprünglichen Einstellungen zurück!

Lösung 2: Weiterleitungen erneuern

Ist das Problem damit immer noch nicht gelöst, ist wahrscheinlich die **.htaccess**-Datei fehlerhaft. Lade die Datei herunter und gib dort folgende Zeilen ein, damit WordPress die URLs korrekt generieren kann:

```
# BEGIN WordPress
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>
# END WordPress# BEGIN WordPress
```

Tipp: Wenn Du Dir nicht sicher bist, welche URL zu einer Fehlerseite führen, hilft Dir unser kostenloser **›Linkchecker**. Auch mit dem Plugin **›Redirection** kommst Du fehlerhaften URLs schneller auf die Spur.

»Fatal error undefined function is_network_admin()«

Wenn Du diese Fehlermeldung siehst, geht erst mal nichts mehr. Doch keine Sorge, mit den folgenden Schritten bringst Du Dein WordPress wieder zum Laufen.

Ursache: Fehlerhaftes Update

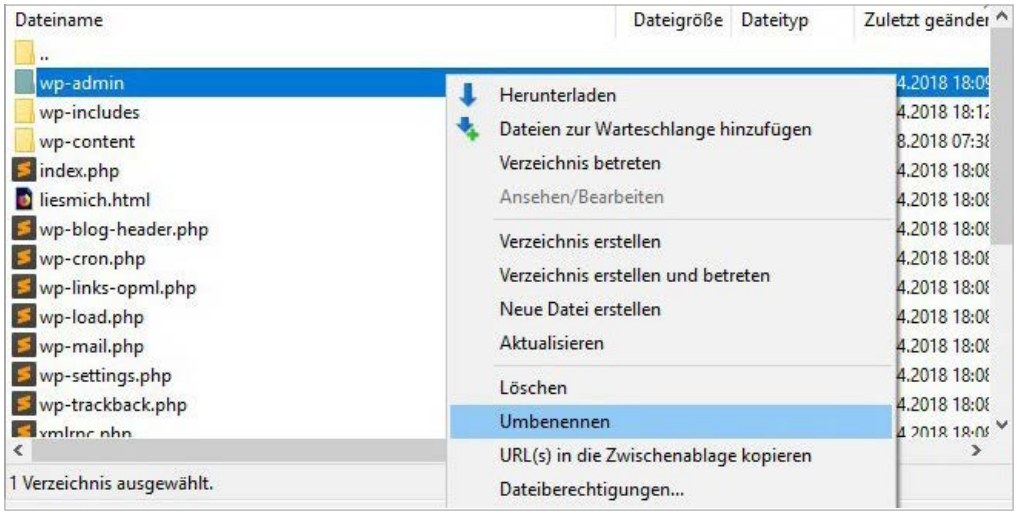
Die Fehlermeldung »Fatal error undefined function is_network_admin()« kann auftreten, wenn Du Dich nach einem automatischen WordPress-Update versuchst einzuloggen. Das Update ist in diesem Fall fehlgeschlagen und Deine Installation beschädigt.

Lösung: Dateien und Verzeichnisse austauschen

1. Erstelle als Erstes ein Backup Deiner WordPress-Installation. Wie das funktioniert, haben wir **›hier beschrieben**.

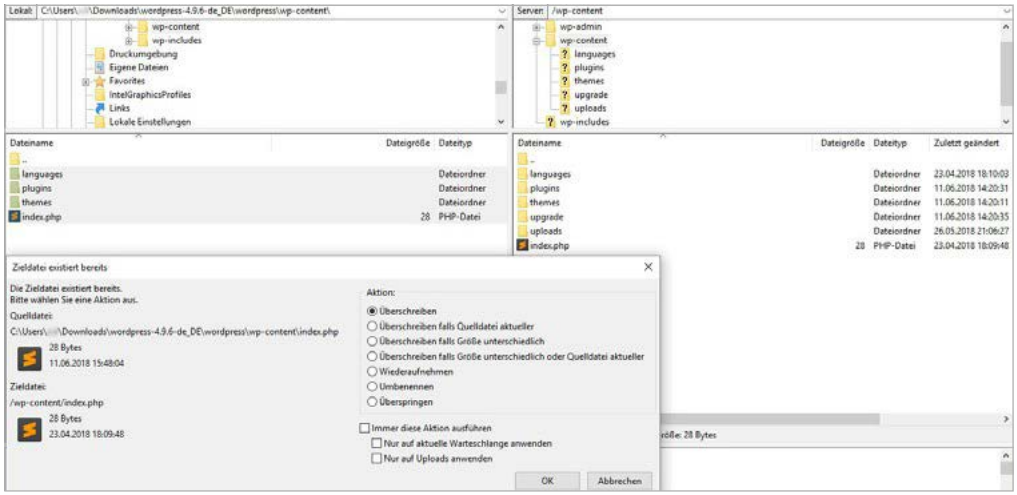
2. Lade WordPress **›in der aktuellen Version** herunter und entpacke die .zip-Datei.

3. Benenne per FTP die Ordner **wp-admin** und **wp-includes** im Hauptverzeichnis Deiner WordPress-Installation um (z. B. mit dem Anhang .bak). Klicke dazu mit der rechten Maustaste auf das entsprechende Verzeichnis und wähle dann **Umbenennen**.



4. Lösche anschließend die Ordner **wp-admin** und **wp-includes** aus dem Hauptverzeichnis und lade diese beiden Ordner aus dem gerade heruntergeladenen WordPress hoch.

5. Lade dann das Verzeichnis **wp-content** hoch und ersetze sämtliche alte Dateien durch die neuen. Deine Child-Themes und Plugins bleiben dabei erhalten. Lösche darüber hinaus keine weiteren Dateien oder Ordner.



6. Lade dann noch alle Dateien auf der Ebene des Hauptverzeichnisses hoch. Achte darauf, nicht noch einmal Verzeichnisse mit hoch zu laden. Keine Sorge: Deine **wp-config.php** wird dabei nicht verändert.

7. Prüfe, ob in der **wp-config-sample.php** neue Einstellungen existieren, die Du Deiner **wp-config.php** hinzufügen möchtest.

8. Lösche die Datei **.maintenance**.

9. Logge Dich in Dein WordPress ein und folge den weiteren Anweisungen.

Wenn Du ein Caching-Plugin verwendest, musst Du abschließend den Cache leeren.

»Error establishing a database connection«

WordPress benötigt für den Betrieb eine mySQL-Datenbank. Darin werden Deine Texte gespeichert und beim Aufrufen sofort angezeigt. Kann die Verbindung nicht hergestellt werden, siehst Du stattdessen diese Fehlermeldung:

Ursachen: Falsche Zugangsdaten, Datenbank defekt/gelöscht/nicht erreichbar

Es kommen verschiedene Ursachen infrage, warum WordPress keine Verbindung zur Datenbank aufbaut: Die Zugangsdaten können falsch sein, vielleicht ist die Datenbank defekt oder der Datenbank-Server nicht erreichbar. Möglich ist auch, dass die Datenbank gelöscht wurde.

Lösung 1: wp-config.php korrigieren

Alle Informationen zur Datenbank findest Du im **›STRATO Kunden-Login** unter **Paket > Datenbanken und Webspace > Datenbankverwaltung**. Prüfe, ob Datenbankname und Benutzername mit den Einträgen in der Datei **config.php** übereinstimmen. Diese Datei liegt im Hauptverzeichnis Deiner WordPress-Installation. Als Serveradresse muss **rdbms.strato.de** (statt **localhost**) eingetragen

sein. Wenn alle Einträge korrekt sind, ändere das Passwort und trage das neue Passwort in die **config.php** ein.

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', '[redacted]');

/** MySQL database username */
define('DB_USER', '[redacted]');

/** MySQL database password */
define('DB_PASSWORD', '[redacted]');

/** MySQL hostname */
define('DB_HOST', 'rdbms.strato.de');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

MySQL-Einträge in der wp-config.php

In der Regel sollte das Problem damit gelöst sein. Andernfalls ist der Datenbank-Server womöglich temporär nicht erreichbar. Das kann zum Beispiel passieren, wenn Deine Website sehr viele Besucher hat und der Server überlastet ist. Mit einem **›Webanalyse-Programm** wie Google Analytics erfährst Du in Echtzeit, wie viele Besucher Deine Website hat.

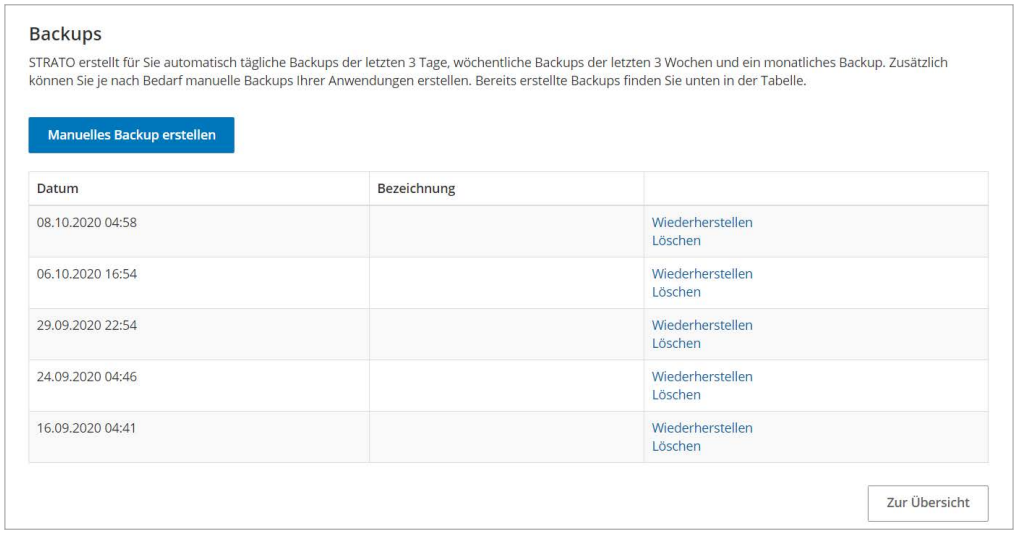
Lösung 2: Datenbank-Backup wiederherstellen

Wenn der bisher beschriebene Weg nicht funktioniert, kannst Du ein (hoffentlich vorhandenes) Datenbank-Backup wiederherstellen. Wie Du Backups anlegst, haben wir **›hier beschrieben**.

STRATO erstellt für Dich automatisch tägliche Backups der letzten drei Tage, wöchentliche Backups der letzten drei Wochen und ein monatliches Backup. Zusätzlich kannst Du je nach Bedarf manuelle Backups Deiner Anwendungen erstellen.

In der Zwischenzeit hinzugefügte oder veränderte Inhalte und Kommentare von Besuchern Deines Blogs gehen bei der Wiederherstellung unwiederbringlich verloren!

Wenn Du WordPress über WordPress & Co. installiert hast, kannst Du das Tool **›STRATO BackupControl** nutzen. Klicke dazu im Kunden-Login auf **Dein Paket > Sicherheit > STRATO BackupControl > Anwendungen (WordPress & Co.)**. So gelangst Du auf die Übersichtsseite Deiner Installationen von WordPress & Co. Um auf die Detailseite einer Installation zu kommen, klickst Du auf **Details & Funktionen**. Unter dem Punkt **Backups** findest Du eine Tabelle mit Deinen Backups. Erstelle dann ein manuelles Backup oder wähle eines der bereits automatisch erstellten Backups aus. Klicke auf **Wiederherstellen**, um die jeweilige Version Deiner WordPress-Installation wiederherzustellen. Mit einem Klick versetzt das Tool Deinen Blog auf den entsprechenden Stand zurück.



5.4 Plugins als Sicherheitsrisiko

Leider sind Plugins für WordPress nicht nur nützlich bis unentbehrlich – sie sind auch das größte Sicherheitsrisiko. Denn es gibt nahezu keine Qualitätskontrolle der angebotenen Plugins und die Qualifikationen der Programmierer in Hinblick auf Sicherheit variieren stark, ohne dass Du das als Anwender erkennen kannst.

Deshalb solltest Du sowohl aus Performance- als auch Sicherheitsgründen so wenig Plugins wie möglich verwenden und auf die Funktionalitäten reduzieren, die wirklich unverzichtbar sind.

Allgemeine Tipps, um das Plugin-Risiko zu minimieren:

- Prüfe neue Plugins möglichst genau. Positive Indizien sind hohe Nutzerzahlen, gute Bewertungen, wenige ernsthafte Probleme in den Support-Foren und regelmäßige Aktualisierungen des Plugins.

- Je umfassender die Funktionalität eines Plugins ist, desto wahrscheinlicher schleichen sich auch bei guten Programmierern Sicherheitslücken ein. Vermeide deshalb Plugins mit zahlreichen Funktionen, die Du größtenteils ohnehin nicht nutzt.

- Installiere generell keine Plugins, die dem User irgendwelche Datei- oder Bilder-Uploads erlauben.

- Lösche unbenutzte Plugins. Deaktivieren genügt nicht.

6 Performance

6.1 WordPress Beine machen per .htaccess

Plugins, Themes, Code Snippets – all das erleichtert den Blogger-Alltag, kann WordPress aber langsam machen. Mit der Folge, dass Besucher vorzeitig abspringen und Suchmaschinen Deine Website mit schlechteren Positionen abstrafen. Im Folgenden zeigen wir Dir, wie Du die Performance von WordPress über die **.htaccess**-Datei verbesserst

1. htaccess-Datei herunterladen

Die **.htaccess**-Datei ist eine Konfigurationsdatei für Webserver. Damit kannst Du unter anderem das Browser-Caching konfigurieren und die Komprimierung aktivieren. WordPress erstellt die **.htaccess** bei der Installation automatisch und legt sie im Hauptverzeichnis ab. Lade die Datei mit Deinem FTP-Programm herunter und öffne sie mit einem Text-Editor.

> Speichere die Original-Version, um im Notfall den alten Zustand wiederherstellen zu können.

2. Browser-Caching nutzen

Beim Browser-Caching legt der Browser Kopien von CSS-, JavaScript- und Grafikdateien im lokalen Speicher ab. Wenn ein Besucher die Seite erneut aufruft, muss der Browser diese nicht erneut vom Server herunterladen. Das spart Traffic und Ladezeit, heißt aber auch: Der Browser eines wiederkehrenden Besuchers aktualisiert zwischenzeitig geänderte Dateien erst nach Ablauf der eingestellten Frist. In den meisten Fällen sollte das aber kein Problem darstellen, sodass Du großzügige Werte (z. B. einen Monat) verwenden kannst.

Das Browser-Caching konfigurierst Du in der **.htaccess**-Datei. In der Regel stehen dort folgende Zeilen:

```
# BEGIN WordPress
# Die Anweisungen (Zeilen) zwischen `BEGIN WordPress` und `END WordPress`
sind
# dynamisch generiert und sollten nur über WordPress-Filter geändert werden.
# Alle Änderungen an den Anweisungen zwischen diesen Markierungen werden
überschrieben.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>

# END WordPress
```

Eigenen Code kannst Du einfach darunter eintragen. Mit der Anweisung **ExpiresByType** gibst Du für die verschiedenen Dateitypen an, wie lange die lokalen Kopien gespeichert werden. Im nachfolgenden Code-Beispiel beträgt die Dauer jeweils einen Monat.

```
# Browser Caching
<IfModule mod_expires.c>
ExpiresActive On
ExpiresByType image/gif „access plus 1 months“
ExpiresByType image/ico „access plus 1 months“
ExpiresByType image/jpeg „access plus 1 months“
ExpiresByType image/jpg „access plus 1 months“
ExpiresByType image/png „access plus 1 months“
ExpiresByType text/css „access plus 1 months“
ExpiresByType text/javascript „access plus 1 months“
ExpiresByType application/x-javascript „access plus 1 months“
ExpiresByType application/javascript „access plus 1 months“
</IfModule>
```

Nach dem Upload der .htaccess kannst Du beispielsweise mit **›gtmetrix.com** prüfen, ob Deine Anweisungen funktionieren. Das Tool schlägt in dem Fall nicht mehr vor, Browser-Caching zu nutzen.

3. Komprimierung aktivieren

Die Komprimierung ist ebenfalls unkompliziert umzusetzen. Zur Methode: Das Programm Gzip komprimiert HTML- und CSS-Dateien. Dadurch ist die von einer Website zu übertragene Datenmenge kleiner. Sobald der Browser die Dateien empfangen hat, entpackt er sie wieder. Weil das Zeit spart, solltest Du die Technik für Deine Website nutzen. Füge dazu einfach die folgenden Zeilen in die **.htaccess** ein:

```
# Gzip Compression
<IfModule mod_deflate.c>
AddOutputFilterByType DEFLATE application/javascript
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
AddOutputFilterByType DEFLATE application/x-font
AddOutputFilterByType DEFLATE application/x-font-opentype
AddOutputFilterByType DEFLATE application/x-font-otf
AddOutputFilterByType DEFLATE application/x-font-truetype
AddOutputFilterByType DEFLATE application/x-font-ttf
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE font/opentype
AddOutputFilterByType DEFLATE font/otf
AddOutputFilterByType DEFLATE font/ttf
AddOutputFilterByType DEFLATE image/svg+xml
AddOutputFilterByType DEFLATE image/x-icon
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/shtml
AddOutputFilterByType DEFLATE text/javascript
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/xml
</IfModule>
```

Nach dem Hochladen kannst Du noch mal das oben genannte Tool nutzen, um zu prüfen, ob der Server die angegebenen Dateitypen tatsächlich komprimiert.

Einfach schneller

Das Warten hat ein Ende. Mit den grundlegenden Tipps in unserem E-Book **›WordPress für Einsteiger** (Kapitel 5.4) und den hier gezeigten Methoden sind kurze Ladezeiten garantiert. Und Deine mobilen Besucher dürften sich neben der Geschwindigkeit auch über das gesparte Datenvolumen freuen.

7 Tracking

7.1 Mehr Sichtbarkeit mit strukturierten Daten

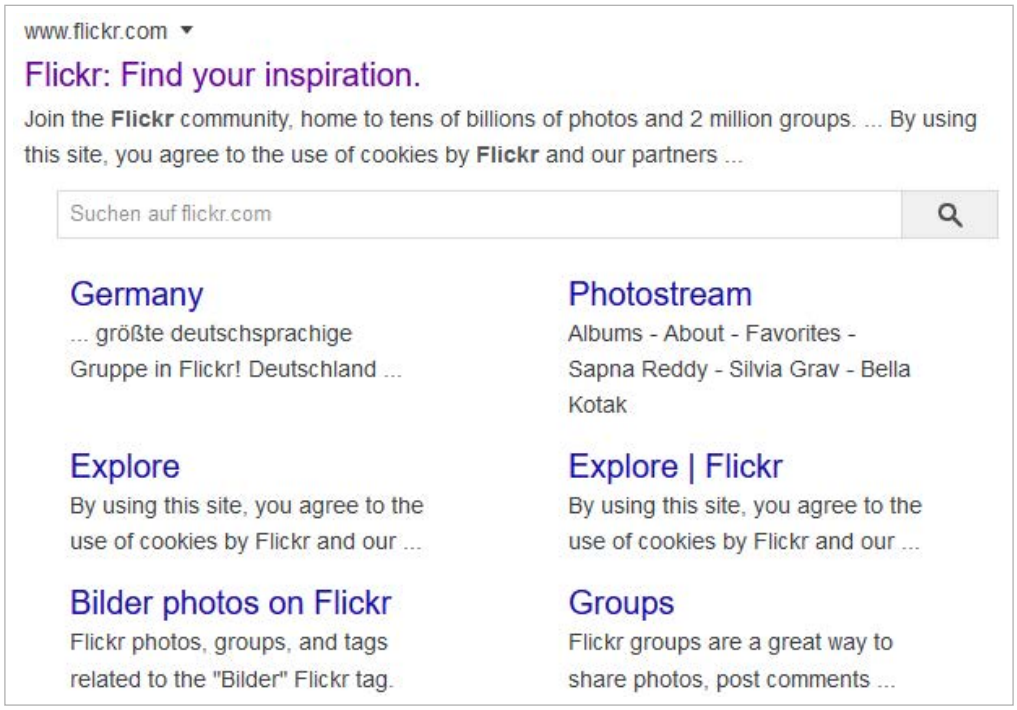
Mit strukturierten Daten erhöhst Du die Sichtbarkeit Deiner Website in den Suchergebnissen. Der Vorteil: Google & Co. präsentieren Deine Inhalte in den Suchergebnissen als sogenannte Rich Snippets, das heißt mit zusätzlichen Informationen wie Bewertungen und Bildern. Bei WordPress ist die Einbindung per Plugin besonders einfach.

Was sind strukturierte Daten?

Die Initiative **›schema.org** hat einen einheitlichen Standard für die Darstellung von Daten entwickelt, der auf bestehenden Auszeichnungssprachen basiert. Mit entsprechenden Tags kannst Du Suchmaschinen zum Beispiel mitteilen, ob es sich bei einem Inhalt um ein Rezept, eine Bewertung und/oder einen Standort handelt. Google, Bing und andere Dienste interpretieren die Daten auf diese Weise besser und zeigen sie in den Suchergebnissen dem Typ entsprechend an.

Rich Snippets für »Erdbeerkuchen Rezept«: Google zeigt Rezepte mit einem Vorschaubild an.

Die Darstellung Deiner Inhalte in den Suchergebnissen kannst Du nicht nur über einen Datentyp beeinflussen. Wie im zweiten Erdbeerkuchen-Beispiel oben zu sehen ist, lassen sich verschiedene Datentypen wie recipe (Rezept), breadcrump (Brotkrumennavigation) und rating (Bewertung) miteinander verbinden. Du kannst auch Sitelinks erstellen und sogar eine Suchfunktion einbinden, welche die Suche direkt auf Deiner Website ausführt (statt in der Suchmaschine).



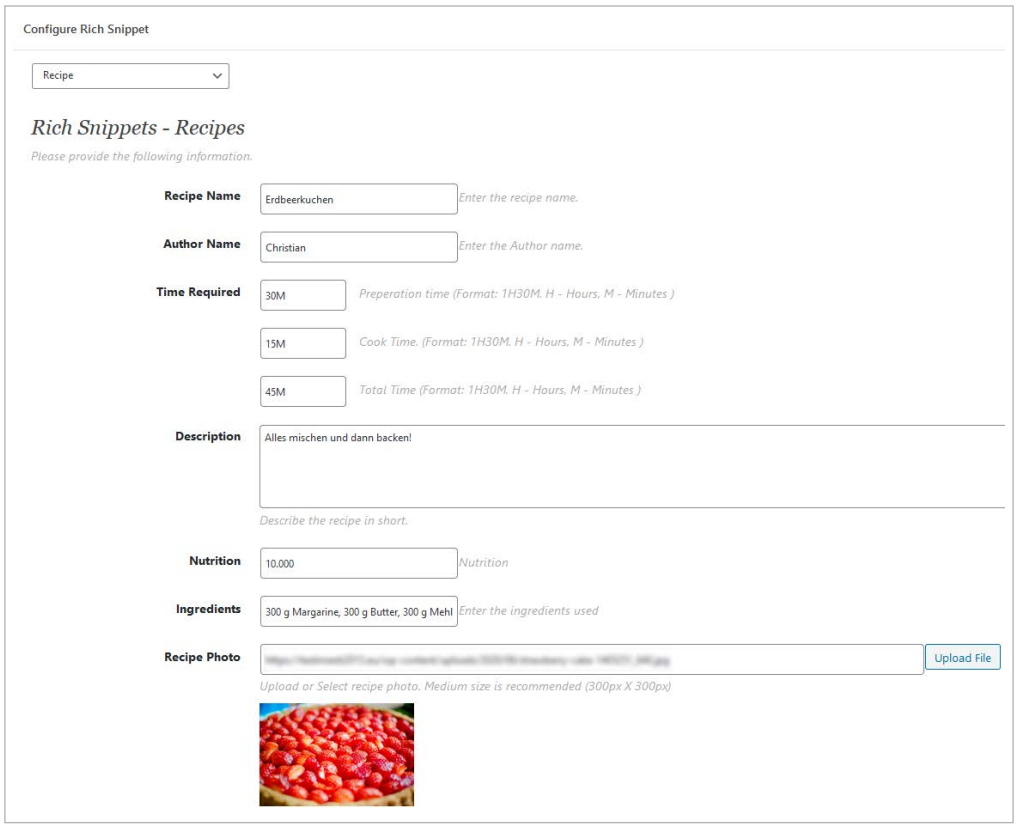
Wie nutze ich strukturierte Daten in WordPress?

Du kannst das schema.org-Markup manuell einpflegen oder den **›Google Tag Manager** nutzen. Letztere Variante ist bequemer, allerdings nur mit Google kompatibel. Am einfachsten ist die Einbindung per Plugin.

Empfehlenswert ist hier zum Beispiel **›Schema – All In One Schema Rich Snippets**. Die Optionen in der kostenlosen Version sind zwar begrenzt, doch immerhin kannst Du Deine Beiträge und Seiten je einem Datentypen zuordnen. Aktuell sind das folgende:

- Item Review
- Events
- Person
- Product
- Recipe
- Software Application
- Video
- Article
- Service

Nach Installation und Aktivierung des Plugins erstellst Du einen neuen Inhalt oder bearbeitest einen bestehenden. Unterhalb des Editors findest Du nun den Bereich **Configure Rich Snippet**. Um beim Beispiel des Erdbeerkuchens zu bleiben: Wähle einfach den Typ **Recipe** und fülle die Felder aus. Achte bei **Time Required** auf das richtige Format und lade zum Schluss ein Vorschaubild hoch:



Teste Deine Einträge nach dem Speichern mit dem **›Testtool für strukturierte Daten**. Im horizontalen WordPress-Menü oben findest Du dazu den Punkt **Test Rich Snippets**. Das Ergebnis sollte fehlerfrei sein, wird aber einige Warnungen enthalten. Das ist kein Grund zur Sorge – das Plugin unterstützt nur nicht alle Tags von schema.org.

Nach dem Speichern erscheint unterhalb des Beitrags bzw. der Seite eine Box mit einigen der Daten, die Du oben eingegeben hast. Die eingebaute Bewertungsfunktion ist sofort einsatzbereit. In den Plugin-Einstellungen kannst Du das Layout der Box an Deine Website anpassen. Die Positionierung lässt sich per CSS über die ID snippet-box festlegen:

```
#snippet-box {
position: relative;
margin: auto;
}
```

Rich Snippets erhöhen die Sichtbarkeit

›Rich Snippets nehmen mehr Platz ein, wirken aufgeräumter und werden daher häufiger geklickt als Standard-Suchergebnisse. Durch die optimierte Darstellung wissen die Nutzer eher, was sie hinter einem Link erwartet. Das erhöht die CTR (Klickrate) und senkt die Absprungrate. Indirekt können strukturierte Daten daher einen positiven Einfluss auf das Ranking Deiner Website haben. Ihr Einsatz lohnt sich also!

Wenn Du strukturierte Daten implementierst, werden Deine Inhalte nicht sofort als Rich Snippets ausgeliefert. Der Zeitpunkt hängt davon ab, wann der Crawler – ein Suchmaschinen-Bot, der automatisch Deine Website durchsucht – das nächste Mal vorbeischaut. Wenn Du regelmäßig interessante Inhalte veröffentlichst, musst Du nicht lange warten. Mehr zum Thema Suchmaschinenoptimierung findest Du **›hier**.

7.2 An diesen Zahlen erkennst Du, ob Dein Blog erfolgreich ist

Zahlen und Statistiken verraten Dir, was auf Deinem Blog gut läuft und was nicht. Mit diesem Wissen kannst Du besser einschätzen, was Deine Leser wollen – aber auch, auf welchem Weg sie Dein Blog überhaupt finden und wie Du sie besser erreichen kannst. Wenn Du Kooperationen mit Firmen suchst oder Werbebanner akquirieren willst, fragen potenzielle Kunden meist nach einigen Erfolgskennzahlen, um einschätzen zu können, ob sich eine Zusammenarbeit lohnt.

Bei Statistiken kommt es wie immer aber darauf an, aus einem Berg von Informationen die wirklich relevanten Zahlen herauszufischen und die Werte richtig zu interpretieren.

Nicht absolute Zahlen, sondern Trends sind wichtig

Hohe, absolute Zahlen sind schön, zeigen aber nur den aktuellen Stand. Zum Messen Deines Erfolgs sind Trends und Entwicklungen viel interessanter. Sie eröffnen Dir nämlich den Blick in die Zukunft: Kommen neue Leser hinzu? Werden sie zu Stammlesern? Steigt die Zahl der Kommentare, Shares und Likes zu Deinen Beiträgen? Kurz: Bist Du auf dem richtigen Weg?

Die periodisch schwankenden Besucherzahlen in diesem Beispiel zeigen, dass an Wochenenden weniger los ist.

Trends und Entwicklungen sind aber auch deshalb interessanter, weil sie sich gut mit denen anderer Blogs vergleichen lassen. Der Analyse-Dienst **›SimilarWeb** liefert Zahlen zu monatlichen Besucherzahlen Deiner und fremder Websites im Zeitverlauf – ideal für die Konkurrenzanalyse. Absolute Zahlen sind dagegen stark abhängig von Deinem Thema, eventuell auch von der Jahreszeit und anderen, individuellen Faktoren.

Dennoch brauchst Du als Grundlage zunächst einmal absolute Zahlen, die Du aus Services wie dem kostenlosen Google Analytics oder auch dem Open-Source-Tool Matomo beziehen kannst.

1. Visits und Unique Visits, Sitzungen und Nutzer

Für den Traffic gibt es drei grundlegende Kennzahlen, die üblicherweise monatlich zusammengefasst werden:

- Seitenaufrufe (Page Impressions): Hier zählt jede einzelne Seite, die Deine Besucher aufrufen.
- Sitzungen (Visits): Alle Seiten, die ein einzelner Besucher zusammenhängend aufruft, ergeben zusammen eine Sitzung.
- Nutzer (Unique Visitors) gibt an, wie viele unterschiedliche Menschen Dein Blog in einem ausgewählten Zeitraum besuchen.

Die Zahl der Seitenaufrufe für sich genommen ist ziemlich uninteressant, schon weil sich das leicht manipulieren lässt. Mehr dazu im Absatz »Aussagekraft statistischer Werte«.

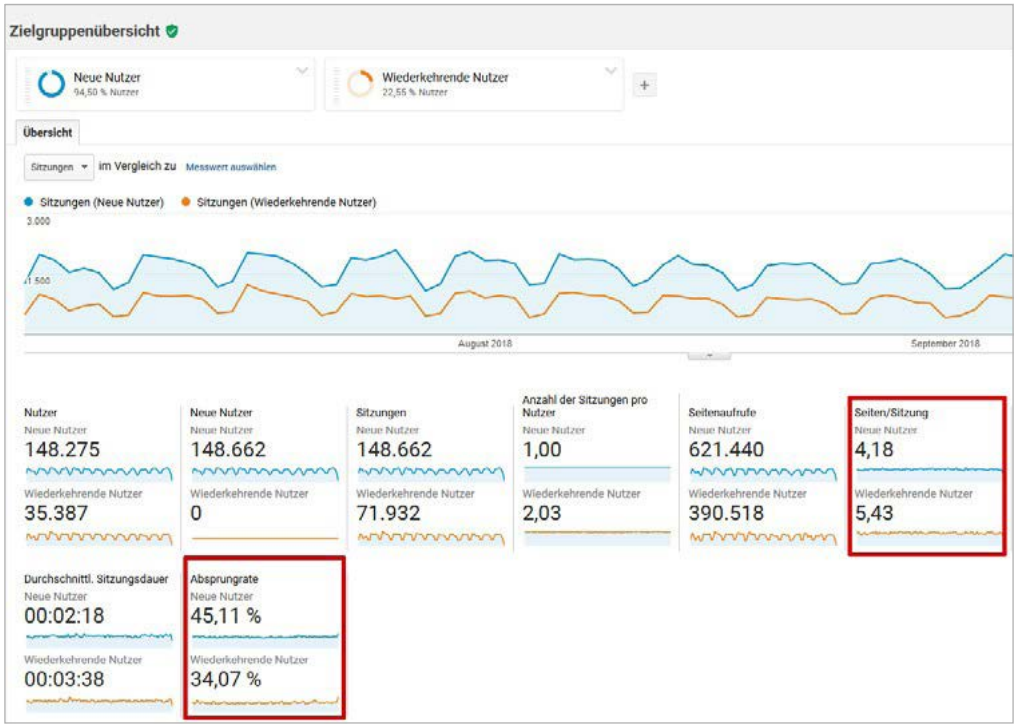
Die viel härtere Währung ist die Zahl der Nutzer. Abgesehen von statistischen Ungenauigkeiten zeigt Dir diese Zahl, wie viele Menschen Du tatsächlich erreichst.

Übersicht der Traffic-Kennzahlen bei Google Analytics

2. Eintagsfliege oder Lesermagnet?

Eine bessere Aussage über die Qualität Deines Blogs liefert die Zahl der Seiten, die ein Besucher im Durchschnitt ansieht. Das sind übrigens überraschend wenig. Wenn Du einen Durchschnittswert von drei oder mehr erzielst, kannst Du stolz sein. Paradoxerweise ist dieser Wert tendenziell niedriger, wenn Du viele Stammleser hast. Denn Stammleser lesen nur den jeweils neuesten Beitrag, weil sie den Rest schon kennen.

Um herauszufinden, ob Dein Blog attraktiv für neue Leser ist, solltest Du die Analyse der Seiten pro Besuch auf neue Leser beschränken. In Google Analytics kannst Du das über **Zielgruppe > Übersicht** einstellen und auch die unterschiedlichen Werte miteinander vergleichen.



Seiten pro Sitzung und Absprungrate von Stammlesern und neuen Besuchern im Vergleich

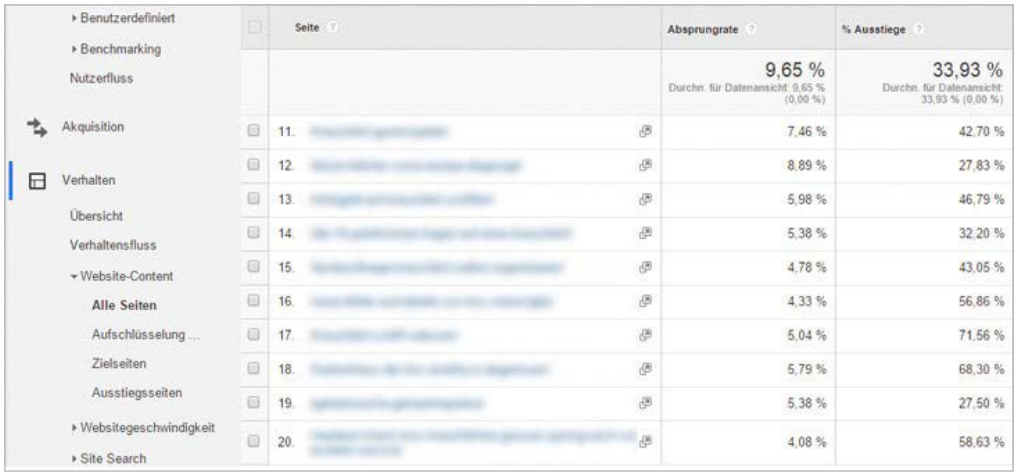
Die **Absprungrate** (engl.: Bounce Rate) hängt eng damit zusammen. Diese Zahl zeigt Dir, wie viele Leser nur eine einzige Seite auf Deinem Blog ansehen und danach wieder gehen. Wie Du beim Thema Stammleser gesehen hast, ist das nicht unbedingt etwas Negatives und kann auch heißen, dass ein Leser, der über Google auf Deiner Seite gelandet ist, genau das gefunden hat, wonach er gesucht hat.

Einen weiteren Anhaltspunkt gibt die **durchschnittliche Sitzungsdauer**. Setzt Du die ins Verhältnis zur Zahl der Seiten pro Sitzung, kannst Du Dir ungefähr vorstellen, wie lange die Besucher sich mit einem einzelnen Beitrag in Deinem Blog beschäftigen. Nicht erschrecken: Auch hier sind die durchschnittlichen Werte recht niedrig – es sind eben Durchschnittswerte, die vor allem von Besuchern nach unten gedrückt werden, die irrtümlich auf Deinem Blog landen und deshalb schon nach zwei oder drei Sekunden wieder gehen.

Die Absprungrate ist aber auch ein Indiz dafür, wie gut es Dir gelingt, Leser zum Bleiben zu bewegen, beispielsweise durch weiterführende Links am Ende jedes Beitrags.

Wenn Du Dich tiefer in diesen Aspekt einarbeiten willst, dann analysierst Du die Absprungrate Deiner Beiträge einzeln. In Google Analytics findest Du eine entsprechende Liste unter **Verhalten > Website-Content > Alle Seiten**. Der Wert **Ausstiege** zeigt, welcher Prozentsatz an Besuchern nach dem Aufruf der jeweiligen Seite Deinen Blog verlassen hat. Bei einzelnen Beiträgen können die Ausstiege schon mal bei 80 Prozent liegen. Das passiert etwa, wenn der Beitrag eine bestimmte Frage sehr präzise beantwortet oder weil Du selbst beispielsweise durch eine Link-Liste dort die Ausstiege regelrecht forcierst.

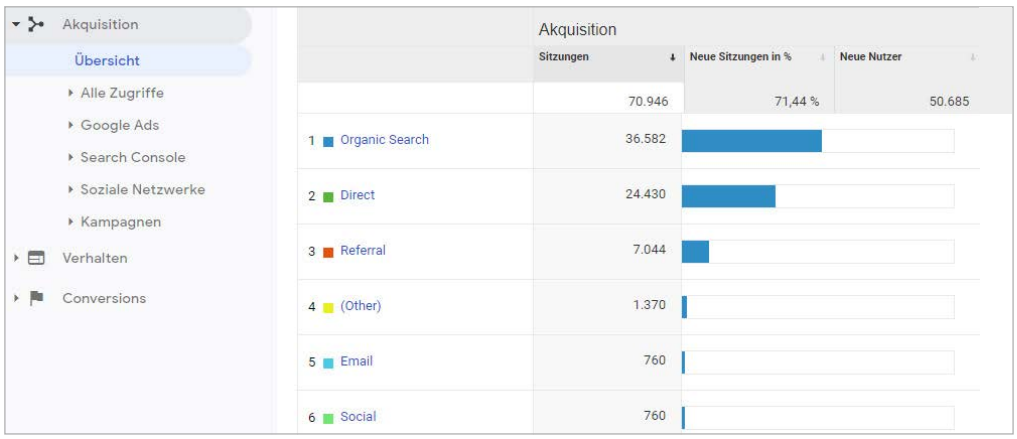
Um Begriffsverwirrungen zu vermeiden: Die Abbruchrate – Bounce Rate – bezeichnet den Anteil der User, die Deinen Blog nach dem Lesen nur einer einzigen Seite wieder verlassen. Die »Ausstiege« zeigen Dir dagegen an, welches die jeweils letzte Seite war, die ein User angesehen hat, bevor er Deinen Blog verlassen hat – unabhängig davon, wie viele Seiten er zuvor bei Dir angesehen hat.



Absprungrate und Ausstiege, aufgegliedert nach einzelnen Blog-Beiträgen.

3. Traffic-Quellen

Jetzt weißt Du, wann Leser gehen. Aber wie haben sie Deinen Blog eigentlich gefunden? Die Herkunft der Besucher ist spannend, weil Du daraus ableiten kannst, wo Du mit mehr Aktivität brachliegende Potenziale ausschöpfen kannst. Und Du kannst identifizieren, welche Kanäle für Deinen Blog besonders gut funktionieren: Ist es eher Facebook oder Twitter? Lohnt sich ein Newsletter oder ein RSS-Feed? Musst Du mehr für Suchmaschinenoptimierung tun, weil zu wenige Besucher über Google zu Dir finden?



Diese Website holt sich viel Traffic über Suchmaschinen.

Die wichtigsten Herkunftsquellen sind:

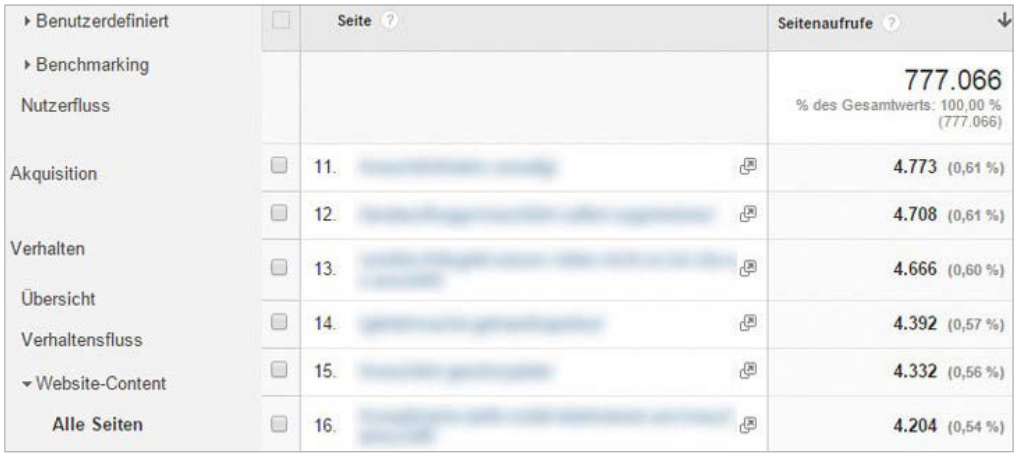
- **Organic Search**: die normale Suche bei Google & Co.
- **Direct**: Bookmarks, direkt eingetippte URLs und ähnliches
- **Referral**: Links von anderen Websites auf Dein Blog
- **(Other)**: zum Beispiel Links aus RSS-Feeds auf
- **EMail**: Links in E-Mails, typischerweise Dein eigener Newsletter
- **Social**: Verweise bei Facebook, Twitter und Co

Mit einem Klick auf den jeweiligen Kanal zeigt Google Analytics Dir weitere Details, etwa von welchen Social-Media-Plattformen genau Deine Besucher kommen.

4. Populäre Texte

Eine der wichtigsten Fragen: Welche Blog-Beiträge waren die beliebtesten? In der bereits für die Ausstiege relevanten Tabelle findest Du bei Google Analytics auch die absoluten Abrufzahlen und prozentualen Werte zu jedem einzelnen Deiner Beiträge.

Achte aber beim Vergleich mehrerer Beiträge darauf, dass manche vielleicht erst ein paar Tage online sind, andere dagegen schon Monate. Letztere haben dadurch automatisch mehr Zugriffe als die ganz frischen.



Analyse der am häufigsten aufgerufenen Blogbeiträge

Konzentriere Dich bei der Auswertung nicht nur auf das Thema des jeweiligen Beitrags als Ursache für dessen Popularität, sondern betrachte auch die Wirkung der Überschrift und des Vorspanns sowie die Textlänge und optische Präsentation, um daraus für künftige Beiträge zu lernen.

Tipp: Wenn Du tiefer in die Analyse einzelner Beiträge einsteigen willst, liefert das WordPress-Plugin **›Google Analytics Dashboard By Analytify** detaillierte Zahlen zu jedem Beitrag direkt im Backend von WordPress.

5. Messen, was andere über Dich denken

Die vielleicht härteste Währung ist »Engagement«, also die Aktivität Deiner Leser, auch »Involvement« oder »Stickyness« genannt. Haben Dein Blog und Deine blogbezogenen Aktivitäten auf Facebook & Co. eine hohe Zahl an Shares, Likes, Mentions, Retweets und Kommentaren, beschäftigen sich Deine Leser intensiv mit dem, was Du schreibst. Gute Werte im direkten Vergleich zu ähnlichen Websites zeigen, dass Du etwas bewegst und Einfluss hast.

Erneut: Beobachte die Steigerungsraten, statt auf absolute Werte zu schauen. Was helfen Dir langfristig 2.500 Facebook-Fans, wenn keine neuen mehr hinzukommen und kaum jemand auf Deine neuen Postings reagiert? Wenige aktive Fans/Freunde sind bei Facebook mehr wert als eine große Zahl passiver Fans, die Du vielleicht

irgendwann einmal über ein Gewinnspiel akquiriert hast. Letztere bekommen Deine Postings vermutlich ohnehin längst nicht mehr zu Gesicht, weil mangels Interaktivität dieser User mit Deiner Fanseite die Filterfunktion von Facebook zuschlägt.

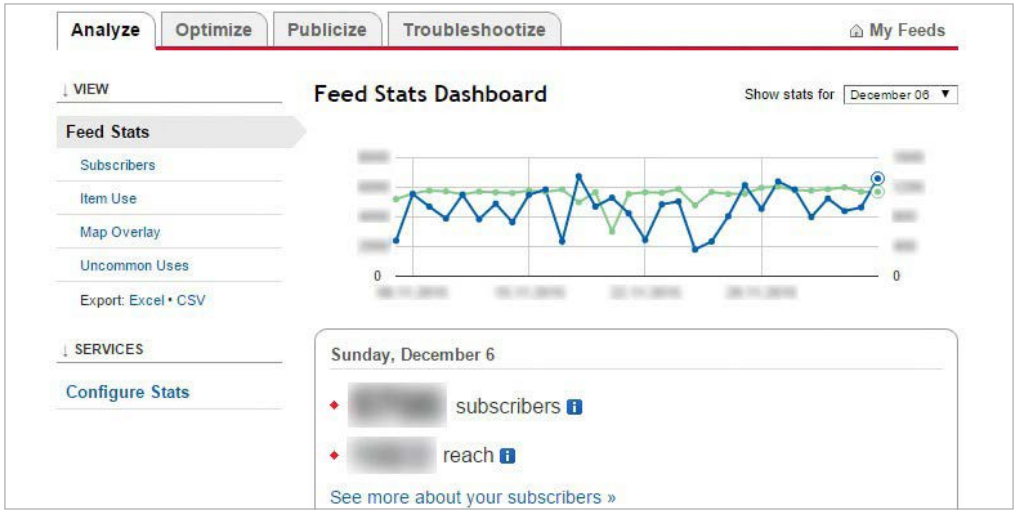
6. Links

Wird Dein Blog von anderen Bloggern wahrgenommen, die sich mit ähnlichen Themen beschäftigen? Ein Anhaltspunkt dafür sind Verbindungen auf Social-Media-Kanälen, ein anderer die eingehenden Links. Die Google Webmastertools beispielsweise liefern Informationen darüber, welche Websites auf Deinen Blog verlinken.

7. Abonnenten

Ein Aspekt, der in Zeiten von Facebook & Co. gerne übersehen wird: Abonnenten Deines Newsletters und RSS-Feeds sind nicht nur wertvolle Indikatoren für die Treue Deiner Leser, sie sind auch eine Traffic-Quelle, die Du direkt in der Hand hast. Denn: Stellst Du einen neuen Beitrag online, erreichst Du diese Abonnenten über Deinen RSS-Feed direkt. Und verschickst Du einen Newsletter, führen die Links darin ohne Umwege auf Deinen Blog.

Damit Du die Zahl Deiner Feed- und Newsletter-Abonnenten messen und deren Reaktion darauf verfolgen kannst, empfiehlt sich die Nutzung von kostenlosen Diensten, die entsprechende Statistiken bereitstellen. Für RSS-Feeds eignet sich **›Feedburner**, für Newsletter lohnt sich ein Blick beispielsweise auf **›Mailchimp**.



Feedburner liefert genaue Statistiken über die Abonnenten Deines RSS-Feeds.

7.3 Content-Gruppen: Traffic detailliert aufschlüsseln mit WordPress und Google Analytics

Im vorigen Kapitel hast Du wichtige Kennzahlen für das Tracking kennengelernt. Mit sogenannten Content-Gruppen gehst Du einen Schritt weiter. Du willst zum Beispiel feststellen, wie oft ein Beitrag vom Typ »News« pro Monat aufgerufen wird, weil ein Anzeigenkunde genau in dieser Kategorie Banner kaufen will. Oder Du willst ermitteln, ob Beiträge mit oder ohne Bildergalerie erfolgreicher sind. Oder Du fragst Dich, wie oft Deine vielen, aufwändig geschriebenen Theaterkritiken überhaupt nennenswert gelesen werden.

Die Möglichkeiten sind vielfältig und das Grundprinzip sehr einfach: Du definierst Regeln, nach denen Google Analytics die Seitenzugriffe gruppieren soll. Anschließend kannst Du die Traffic-Zahlen nach diesen individuellen Kriterien aufschlüsseln.

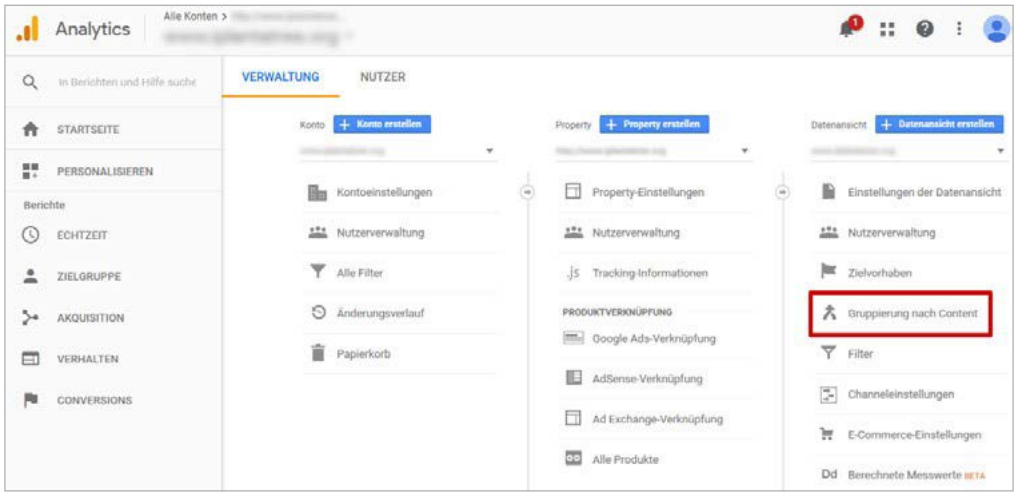
Google Analytics bietet unterschiedliche, teils sehr mächtige, aber auch komplizierte Methoden, um dieses Ziel zu erreichen. Für Blogger sind die sogenannten Content-Gruppen sicherlich die einfachste und bequemste. Je nach Deinen Zielen genügen ein paar Klicks in den Google-Analytics-Einstellungen. Bei komplexeren Anforderungen und für User, die sich ein wenig mit JavaScript und PHP auskennen, lassen sich Content-Gruppen noch individueller einsetzen. Wir beschreiben beide Wege Schritt für Schritt.

Vorarbeit: Bereiche definieren

Zunächst musst Du entscheiden, was genau Du mit den Content-Gruppen erfassen willst. Für die Schritt-für-Schritt-Anleitung entscheiden wir uns für ein einfaches Beispiel: Wir wollen ermitteln, wie viele Zugriffe die Homepage, die Kategorie **News** und die Kategorie **Bildergalerie** jeweils haben. Die neu zu erstellende Content-Gruppe in Google Analytics nennen wir »Meine Bereiche«.

Content-Gruppen in Google Analytics anlegen

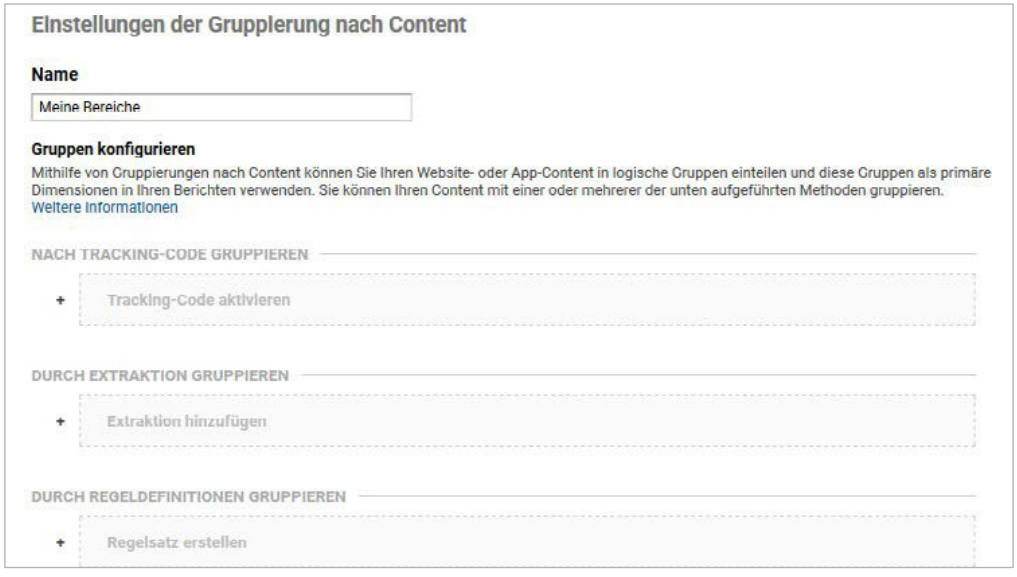
Klicke in Google Analytics links unten auf **Verwaltung** und wähle anschließend die **Datenansicht** aus, für die Du eine Content-Gruppe erstellen willst. Anschließend klickst Du unten auf **Gruppierung nach Content**.



Anlegen einer neuen Content-Gruppe in Google Analytics

Nun klickst Du auf den roten Button **+ Neue Gruppierung nach Content**. Gib Deiner Gruppierung einen aussagekräftigen Namen (in unserem Beispiel »Meine Bereiche«). Jetzt hast Du eine Obergruppe erstellt, von denen Google maximal fünf bereitstellt. Unterhalb einer Obergruppe kannst Du aber beliebig viele Untergruppen anlegen.

VVon hier an unterscheiden sich die Methoden: Die Option **Nach Trackingcode gruppieren** ist – ebenso wie **Durch Extraktion gruppieren** – für erfahrene Nutzer mit PHP- und JavaScript-Kenntnissen gedacht. Darauf gehen wir später noch ein.



Die drei Möglichkeiten, Content-Gruppen in Google Analytics zu definieren

Durch Regeldefinitionen gruppieren ist dagegen ohne Eingriff in den Programmiercode von WordPress möglich und damit am einfachsten anzuwenden.

Gruppieren mit Regeldefinitionen

Klickst Du auf das graue Feld **Regelsatz erstellen**, kannst Du Deine Regeln frei definieren. Mit der Auswahl **Seite** filterst Du nach Elementen in der URI der jeweils aufgerufenen Seite. Die URI (Uniform Resource Identifier) ist der Teil der URL, der nach **http://** und Deiner Domain steht. Die URI von

http://www.katzenfreunde-suedsachsen.de/ernaehrung/tipps.html

ist also

/ernaehrung /tipps.html

Die Auswahl **Seitentitel** bezieht sich dagegen auf den **title**-Tag im HTML-Code der jeweils aufgerufenen Seite. Oft entspricht das dem Titel des jeweiligen Blogbeitrags – aber das ist abhängig vom jeweiligen Theme, sodass Du prüfen musst, welcher Text bei Deinem Theme im **title**-Tag steht. Rufe dazu einfach den Quelltext einer Seite im Browser auf (in den meisten Browsern: **Strg + U**) und suche nach **<title>**. Innerhalb des öffnenden und schließenden Title-Tags findest Du den gesuchten Text:

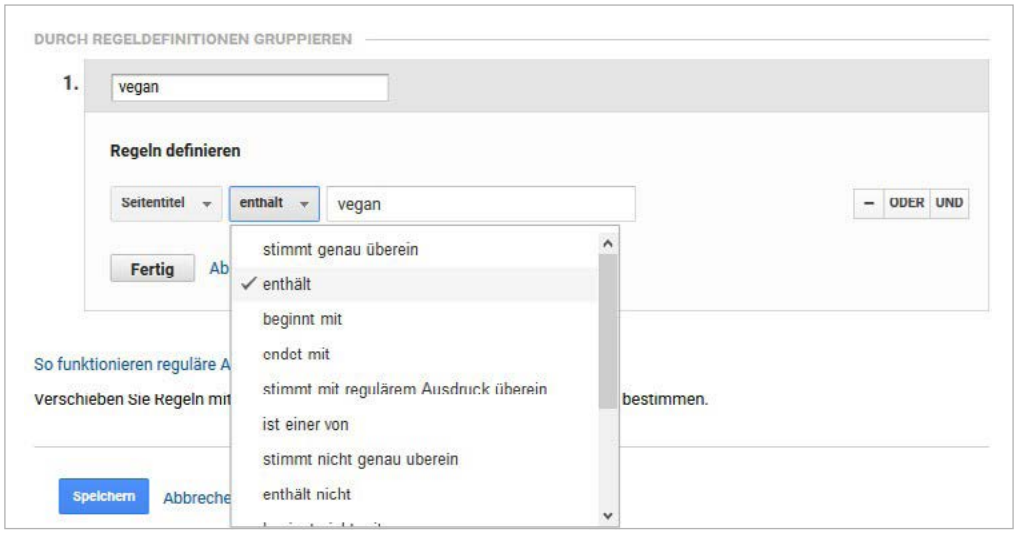
<title>das ist der Text, den Du suchst</title>

```
<!DOCTYPE html>
<html lang="de-DE" class="no-js no-svg">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="profile" href="http://gmpg.org/xfn/11">

<title>Immer mehr Katzen ernähren sich vegan &#8211; Katzenfreunde Südsachsen</title>
<link rel='dns-prefetch' href='//fonts.googleapis.com' />
<link rel='dns-prefetch' href='//s.w.org' />
<link href='https://fonts.gstatic.com' crossorigin rel='preconnect' />
<link rel="alternate" type="application/rss+xml" title="Katzenfreunde Südsachsen &raquo
<link rel="alternate" type="application/rss+xml" title="Katzenfreunde Südsachsen &raquo
<link rel="alternate" type="application/rss+xml" title="Katzenfreunde Südsachsen &raquo
        <script type="text/javascript">
            window._wpemojiSettings = {"baseUrl":"https:\/\/s.w.org\/images\/core\/emo
            !function(a,b,c){function d(a,b){var c=String.fromCharCode;l.clearRect(0,0,
        </script>
        <style type="text/css">
img.wp-smiley,
img.emoji {
    display: inline !important;
```

Der <title>-Tag im HTML-Code eines Blogbeitrags

Du könntest also beispielsweise mit einer **Seitentitel**-Regel, der Option **enthält** und dem Suchtext **»vegan«** alle Blogbeiträge filtern, deren Titel »vegan« enthält.

Beispiel: Gruppiere alle Beiträge, die **»vegan«** enthalten.



Für die Regeldefinition stehen viele Optionen zur Auswahl.

Du kannst natürlich mehrere Definitionen erstellen, um unterschiedliche Gruppen von Inhalten zu differenzieren. Dabei musst Du lediglich darauf achten, dass Analytics die Regeln von oben nach unten durcharbeitet und stoppt, sobald ein Kriterium erfüllt ist. Suchst Du also erst nach einem Titel, der »Katze« enthält, und anschließend nach einem Filter mit »Katzenfutter« im Titel, werden auch alle »Katzenfutter«-Beiträge als »Katze« gezählt, weil bereits die erste Bedingung erfüllt ist.

Auch Kombinationen mehrerer mit **und/oder** verknüpfte Kriterien kannst Du hier definieren.

Mehrere Kriterien verknüpfst Du mit und/oder.

Die Vielzahl der Filtermöglichkeiten kann da schnell unübersichtlich werden und zu logischen Fehlern führen. Prüfe also Deine Regeln sehr genau, damit die Auswertung letztlich sinnvolle Werte ergibt.

Zurück zu unserem Beispiel der Sortierung nach **News**, **Bildergalerie** und **Homepage**. Voraussetzung für diese Methode ist, dass sich die Kategorien über die URI identifizieren lassen. News-Beiträge hätten also beispielsweise die URL **http://www.katzenfreunde-suedsachsen.de/news/testnews.html** und Beiträge der Bildergalerie-Kategorie etwa **http://www.katzenfreunde-suedsachsen.de/bildergalerie/galeriebeitragtest.html**.

Eine entsprechende Content-Gruppierung erreichst Du dann mit folgenden Regeln:

Seite enthält nicht /

Seite beginnt mit /news/

Seite beginnt mit /bildergalerie/

Die einzelnen Regel-Definitionen kannst Du leider in der Übersicht aller Regeln in Google Analytics nicht direkt sehen. Dazu musst Du jede Regel zum Bearbeiten einzeln öffnen.

Der fertige Regelsatz in Google Analytics

Bitte beachte aber, dass die Regeln je nach Konfiguration Deines Webspaces und Deiner WordPress-Installation unterschiedlich ausfallen können. Du solltest daher ausführlich testen und alle Varianten berücksichtigen, die eine Rolle spielen. Beispielsweise könnte Deine Homepage sowohl mit als auch ohne abschließendem »/« aufrufbar sein. In diesem Fall müsstest Du die Regel für die Homepage-Erkennung entsprechend abändern, damit Google Analytics beide Varianten berücksichtigt.

Falls die Filterung über die Regeldefinitionen nicht funktioniert, weil weder URI noch Seitentitel die nötigen Unterscheidungsmerkmale bieten, bleibt nur der aufwändigere Weg über den Tracking-Code.

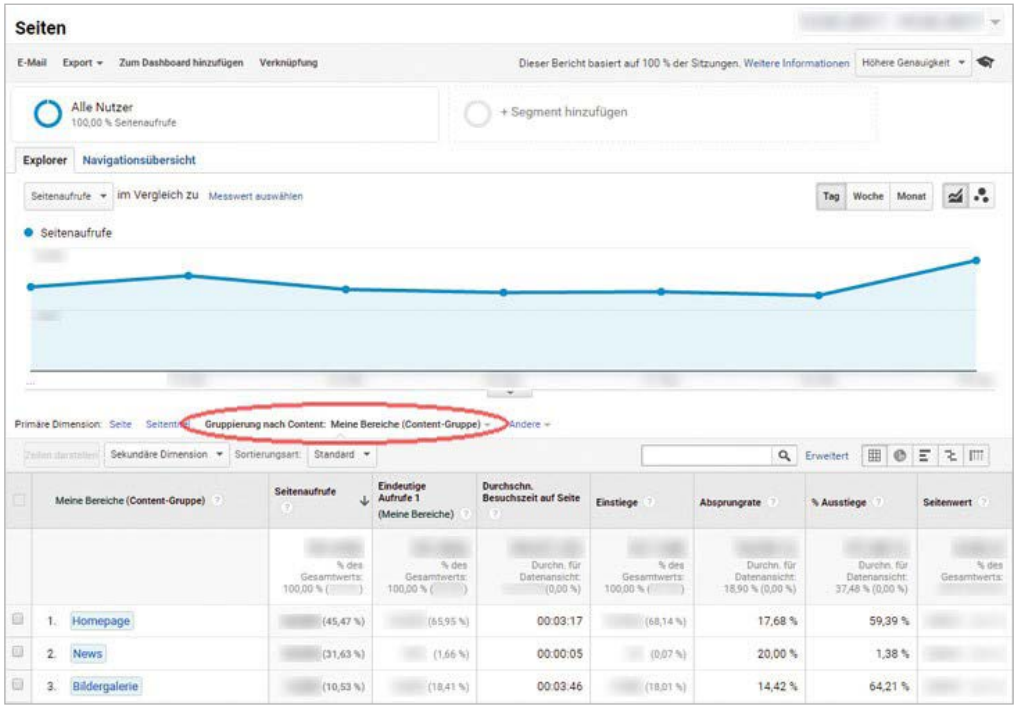
Doch sehen wir uns zunächst an, wie Du die Content-Gruppen bei der Auswertung der Daten von Google Analytics nutzen kannst.

Content-Gruppen bei der Traffic-Analyse anwenden

Wechsle in Google Analytics in die Ansicht **Verhalten > Websitecontent > Alle Seiten**. Zwischen der Grafik und der Wertetabelle findest Du die Option **Gruppierung nach Content**.

Hier wählst Du nun Deine neue angelegte Content-Gruppe »Meine Bereiche« aus. Die Gruppe wird angezeigt, sobald Analytics ausreichend Daten gesammelt hat. Sie ist also nicht sofort nach Erstellen sichtbar. Spätestens am folgenden Tag sollten aber genügend Daten vorhanden sein.

Jetzt zeigt Google Analytics die Traffic-Zahlen sortiert nach Deinen selbst definierten Content-Bereichen an.



So sieht die Auswertung Deiner Content-Gruppen in Google Analytics aus.

Content-Gruppen via Trackingcode

Die zweite Methode zum Erstellen von Content-Gruppen setzt zumindest minimale Kenntnisse in JavaScript, PHP und WordPress-Programmierung voraus. Sehr kompliziert ist aber auch diese Methode nicht.

Der Vorteil via Trackingcode: Du bist nicht auf Unterscheidungsmerkmale angewiesen, die in der URI oder im Seitentitel enthalten sind. Stattdessen übergibst Du mithilfe von ein paar Zeilen JavaScript- und PHP-Code in Deinem WordPress-Theme zusätzliche Informationen an Google Analytics. Dort kannst Du die Traffic-Auswertung später genau nach diesen Informationen filtern. Du gibst also quasi jedem Seitenaufruf bei der Zählung eine zusätzliche Markierung mit auf den Weg, die Du dann später zur Auswertung nutzen kannst.

Wie Du im Folgenden sehen wirst, klingt das viel komplizierter, als es ist. Um beispielsweise nach unterschiedlichen WordPress-Kategorien zu filtern, genügt eine einzige Codezeile PHP plus eine Zeile JavaScript im Analytics-Trackingcode.

Child-Theme anlegen und Analytics-Code einbinden

Zunächst empfiehlt es sich, für die notwendigen kleinen Veränderungen im Theme ein Child-Theme anzulegen (**›siehe Kapitel 2.1**).

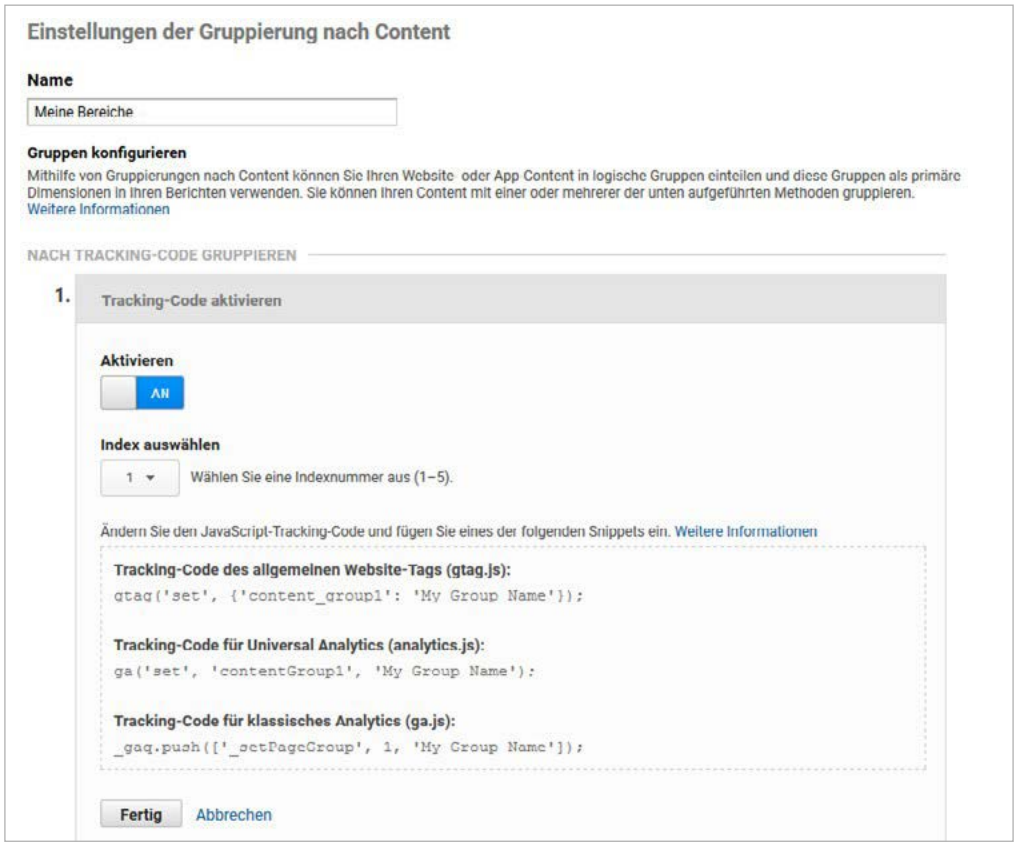
Wenn Du den Google-Analytics-Code bisher über ein Plugin eingebunden hattest, musst Du dieses Plugin deaktivieren und den Code manuell im Theme einbinden. Dazu kopierst Du den Analytics-Code einfach in den Head-Bereich des HTML-Codes. Typischerweise findest Du den Head-Bereich in der WordPress-Datei **header.php**. Füge den Analytics-Code vor dem HTML-Tag **</head>** ein.

Content-Gruppen mit Tracking-Code

Zunächst ist der Ablauf identisch zum Anlegen einer Content-Gruppe mit Regeldefinitionen (siehe oben). Statt auf das graue Feld **Regelsatz definieren** klickst Du jetzt aber auf das Feld **Tracking-Code aktivieren**. Da Du bei dieser Methode den Code direkt veränderst, solltest Du ein Grundverständnis für den Analytics-Tracking-Code mitbringen.

Nun siehst du den Tracking-Code, den Du in Dein Theme einfügen musst. Allerdings werden wir diesen Code im nächsten Schritt noch modifizieren, bevor wir ihn

einfügen. Die Anleitung geht davon aus, dass Du das neue Universal Analytics nutzt, sodass die zweite der aufgeführten Code-Zeilen gilt.



Die ergänzende Zeile für den Tracking-Code von Google Analytics

Klicke auf **Fertig** und dann **Speichern**, um die Content-Gruppe fertig anzulegen.

Parameter in WordPress ermitteln

Nun ist Google Analytics bereit, die Daten für die neu angelegte Content-Gruppe zu empfangen. Damit der entsprechende Parameter an Analytics übergeben wird, musst Du den oben bereits angesprochenen Tracking-Code im WordPress-Theme um eine Zeile erweitern.

Allerdings fehlt Dir dazu noch ein wichtiges Element: die zu übergebenden Daten – in unserem Beispiel also der Name der Kategorie, in dem sich der jeweilige Blogbeitrag befindet. Das ist die Stelle, an der ein wenig PHP-Code ins Spiel kommt.

Geh an die Stelle, an der Du in der **header.php** den Analytics-Code eingefügt hast. Direkt über der Zeile **<script>** fügst Du folgende Zeilen ein:

```
<?php
$meinebereiche='';

if(in_category(,bildergalerie')) {$meinebereiche='Bildergalerie';}
else if(in_category(,news')) {$meinebereiche='News';}
else if(is_home()) {$meinebereiche='Homepage';}
?>
```

Wir bedienen uns hier einer Eigenheit von WordPress, den sogenannten Conditional Tags (› **https://codex.wordpress.org/Conditional_Tags**). Das sind PHP-Funktionen, mit denen Du die Eigenschaften des jeweils aktuellen Blogposts ermitteln kannst, also beispielsweise welche Kategorien oder Schlagwörter einem Beitrag zugeordnet sind, ob es sich um die Homepage, eine Kategorie-Übersicht, eine einzelne Seite oder ein Posting handelt und vieles mehr.

In unserem Bespiel verwenden wir zwei Conditional Tags: **in_category()** und **is_home()**.

Wenn der aufgerufene Beitrag der Kategorie mit der Titelform »bildergalerie« zugewiesen ist, gibt **in_category(‚bildergalerie')** den Wert **true** zurück und erfüllt damit die **if**-Bedingung. Google Analytics gekommt also das Signal, dass es sich um den Aufruf einer Seite im Seitenbereich »Bildergalerie« handelt. Das Gleiche würde auch mit dem Kategorienamen oder der Kategorie-ID funktionieren.

is_home() gibt **true** zurück und erfüllt damit die **if**-Bedingung, wenn es sich bei der aufgerufenen Seite um die Homepage handelt.

Es würde zu weit führen, Conditional Tags von WordPress hier ausführlich darzustellen, aber mit ein wenig Einarbeitung kannst Du mit der Vielzahl von möglichen Tags auch recht komplexe und stark differenzierte Content-Gruppen bilden. Interessant dafür könnten beispielsweise auch Conditional Tags wie **is_autor()** zur Ermittlung des Autors eines Beitrags sein oder **is_single(‚beispieltag')** für Posts mit dem Schlagwort »beispieltag« oder **has_post_thumbnail()**, um zu klären, ob dem Beitrag ein Artikelbild zugeordnet ist.

Ergänzung des Analytics-Codes

Die PHP-Variable **$meinebereiche** enthält in unserem Beispiel jetzt einen der Werte **Bildergalerie**, **News** oder **Homepage** oder ist leer, wenn die aufgerufene Seite keinem der drei Bereiche angehört.

Im nächsten Schritt musst Du den Inhalt der Variable nun noch als Parameter an Google Analytics übergeben. Dazu erweiterst Du den Analytics-Code um die folgende Zeile:

```
ga(‚set‘, ‚contentGroup5 ,, ‚<?=$meinebereiche?>‘);
```

contentGroup5 ist der Parameter-Name, den Google Analytics in dem JavaScript-Code zu Deiner neu angelegten Content-Gruppe definiert hat. Diesen Parameter belegen wir in der Code-Zeile nun mit dem Wert der PHP-Variable **$meinebereiche** und schon erfasst Google Analytics, aus welchem Deiner selbst gewählten Bereiche die gerade erfasste Seite stammt.

Der komplette Code für unser Beispiel lautet (statt UA-XXXXXX-XX gibst Du Deinen individuellen Google Analytics Code ein):

```
<?php
$meinebereiche=‘‘;
if(in_category(‚bildergalerie‘)) {$meinebereiche=‘Bildergalerie‘;}
else if(in_category(‚news‘)) {$meinebereiche=‘News‘;}
else if(is_home()) {$meinebereiche=‘Homepage‘;}
?>
<script>
  (function(i,s,o,g,r,a,m){i[‚GoogleAnalyticsObject‘]=r;i[r]=i[r]||function(){
  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
  m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.
  insertBefore(a,m)
  })(window,document,‘script‘,‘//www.google-analytics.com/analytics.js‘,‘ga‘);
  ga(‚create‘, ‚UA-XXXXXX-XX‘, ‚auto‘);
  ga(‚set‘, ‚contentGroup5 ,, ‚<?=$meinebereiche?>‘);
  <em>ga(‚set‘, ‚anonymizeIp‘, true);
  ga(‚send‘, ‚pageview‘);
</script></em>
```

Jetzt heißt es ein wenig warten, damit Google Analytics Daten sammeln kann, bevor Du sie in der Traffic-Auswertung sehen kannst.

Hast Du alles richtig gemacht, wirst Du die Segmentierung Deines Contents spätestens nach einigen Stunden in Google Analytics sehen. Beachte aber, dass die Gruppierung natürlich nicht rückwirkend funktioniert. Analytics zeigt die Content-Gruppen-Auswertung also erst an, wenn es entsprechende Daten sammeln konnte.

Die Auswertung der gesammelten Daten funktioniert genauso wie oben bereits beschrieben. Du kannst sogar die unterschiedlichen Methoden der Content-Gruppierung kombinieren – Trackingcode, Extraktion (Profi-Option für Filter über sogenannte Regular Expressions) und Regeldefinitionen. Google Analytics arbeitet die drei Varianten in genau dieser Reihenfolge ab und stoppt daher, sobald in diesem Ablauf das erste Kriterium erfüllt ist.

8 WordPress Multisite

Mit **›WordPress Multisite** kannst Du beliebig viele WordPress-Websites in einer Installation zentral verwalten. Themes und Plugins musst Du nur einmal im Netzwerk installieren und kannst sie dann für alle Websites einsetzen. Die Administration ist dadurch viel effizienter. Praktisch ist Multisite auch zum Testen neuer Funktionen oder Themes.

Bei der Einrichtung hast Du die Wahl, ob die Netzwerk-Seiten Subdomains, Verzeichnisse oder verschiedene Domains verwenden sollen. Dies kann später nicht geändert werden.

8.1 Multisite mit Verzeichnissen und Subdomains

Schritt 1: WordPress manuell installieren

Die STRATO Installationshilfe WordPress & Co. eignet sich perfekt für die Installation einzelner Blogs. Für ein Netzwerk von Blogs musst Du WordPress jedoch manuell installieren. Wie das funktioniert, haben wir in **›Kapitel 1: Voraussetzungen** beschrieben.

Schritt 2: Multisite aktivieren

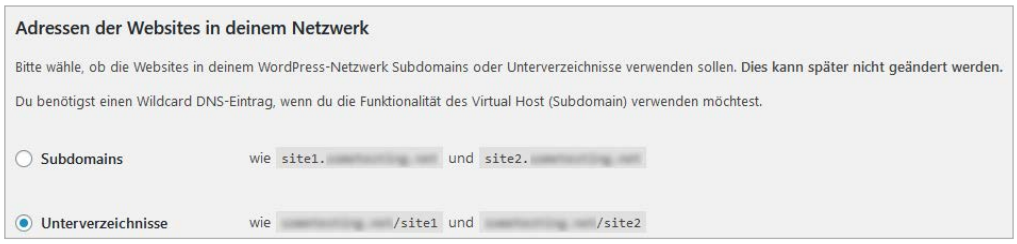
Um Multisite zu aktivieren, musst Du nach der Installation die Datei **wp-config.php** anpassen. Falls diese im Webspace nicht zu sehen ist, hat WordPress sie noch nicht erstellt. Lösung: In WordPress einloggen und dann die FTP-Verbindung neu aufbauen. Lade die **wp-config.php** herunter und gib oberhalb von »/* That's all, stop editing! Happy publishing. */« folgende Zeile ein:

```
define(‚WP_ALLOW_MULTISITE', true);
```

Anschließend lädst Du die Datei wieder hoch und ersetzt die Original-Datei im Webspace.

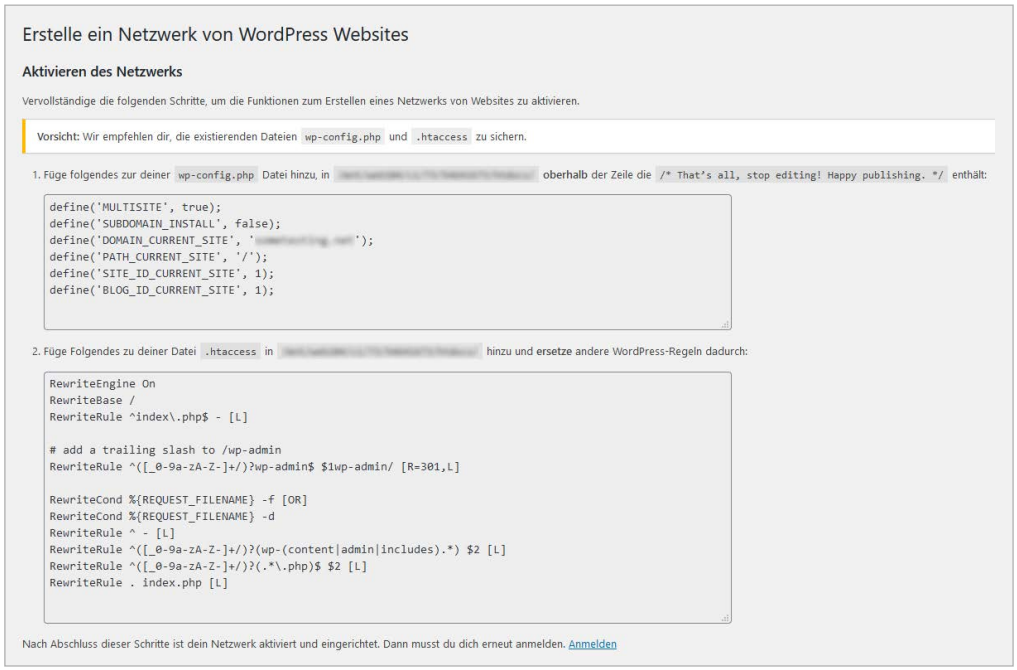
Schritt 3: Netzwerk aktivieren

Wenn Du das Backend aktualisierst, erscheint unter **Werkzeuge** nun der Eintrag **Netzwerk-Einrichtung**. Unter »Adressen der Websites in deinem Netzwerk« kannst Du zwischen Subdomains und Unterverzeichnissen wählen:



Bei beiden Varianten werden nach einem Klick auf den Installieren-Button zwei Code-Schnipsel angezeigt: Der erste wird an der gleichen Stelle wie oben in die **wp-config.php**, der zweite Block in die **.htaccess** eingefügt.

> Bitte sichere beide Dateien vorab und achte darauf, die **.htaccess** im Hauptverzeichnis abzulegen!



Bei der Netzwerk-Einrichtung sagt WordPress, welcher Code wo eingefügt werden muss.

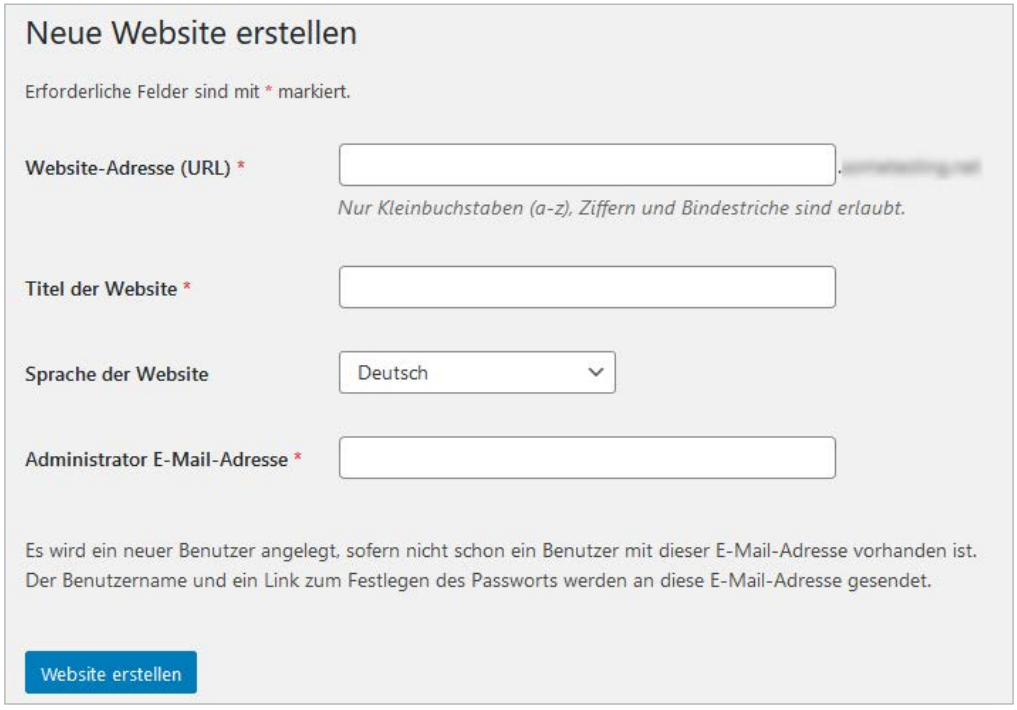
Nach einem erneuten Login ist das Netzwerk fertig eingerichtet. Wenn Du bei der Einrichtung Unterverzeichnisse ausgewählt hast, kannst Du gleich anfangen, die Websites zu erstellen. Klicke dazu oben links neben dem WordPress-Logo auf **Meine Websites > Netzwerkverwaltung > Websites > Neu hinzufügen**.

Schritt 4: Subdomains erstellen

Hast Du Dich für die Option Subdomains entschieden, musst Du im **›STRATO Kunden-Login** erst noch Subdomains für die einzelnen Websites anlegen. Über **Dein Paket > Domains > Domainverwaltung > Verwalten > Subdomain anlegen** ist das schnell erledigt.

Um abschließend die Websites zu erstellen, gehst Du wie bei den Unterverzeichnissen vor: **Meine Websites > Netzwerkverwaltung > Websites > Neu hinzufügen**. Als **Website-Adressen** (URL) gibst Du die oben erstellten Subdomains an.

Wichtig: Um neben der Hauptseite auch die Subdomains per SSL zu verschlüsseln, benötigst Du ein **›Wildcard-Zertifikat**. Dieses kannst Du in Deinem Kunden-Login unter **Sicherheit > STRATO SSL** hinzubuchen.



In der Netzwerkverwaltung kannst Du bequem neue Websites erstellen.

WordPress Multisite: effizient bei vielen Blogs

WordPress Multisite ist ein mächtiges Tool, um beliebig viele Blogs und Websites zu administrieren. Als sogenannter Super-Administrator kannst Du Themes und Plugins zentral installieren, aktualisieren und für einzelne Websites aktivieren. Wird ein Benutzer auf Website-Ebene angelegt, hat er – wie bei separaten Installationen auch – nur dort entsprechende Rechte.

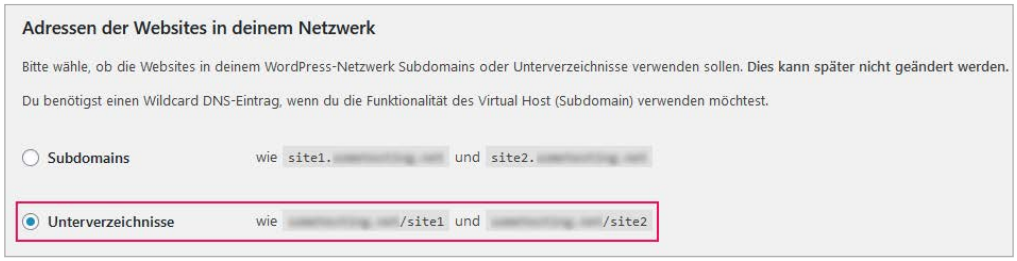
Wie die Anbindung unterschiedlicher **›STRATO Domains** funktioniert, erfährst Du im folgenden Abschnitt.

8.2 Domain Mapping mit WordPress Multisite

Bei WordPress Multisite können Websites des Netzwerks nicht nur Unterverzeichnissen oder Subdomains zugewiesen werden, sondern auch anderen Domains. Für die folgenden Schritte benötigst Du daher ein **›STRATO Hosting-Paket** mit mindestens zwei Domains.

Schritt 1: WordPress installieren und Multisite aktivieren

Als Erstes installierst Du WordPress manuell und aktivierst die Multisite-Funktion und das Netzwerk wie in **›Kapitel 8.1** beschrieben (Schritte 1 bis 3). Wähle im dritten Schritt als Adressen der Websites unbedingt Unterverzeichnisse:

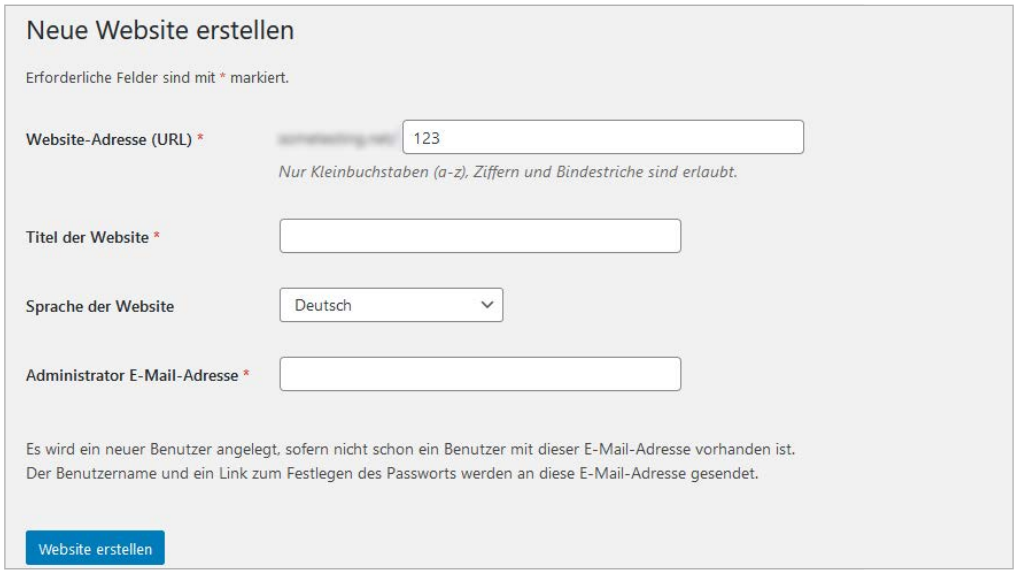


Um Probleme beim Einloggen oder mit Berechtigungen zu vermeiden, fügst Du unterhalb des von WordPress vorgegebenen Codes zusätzlich folgende Zeile in die **wp-config.php** ein:

```
define(‚COOKIE_DOMAIN‘, ‚‘);
```

Schritt 2: Websites hinzufügen

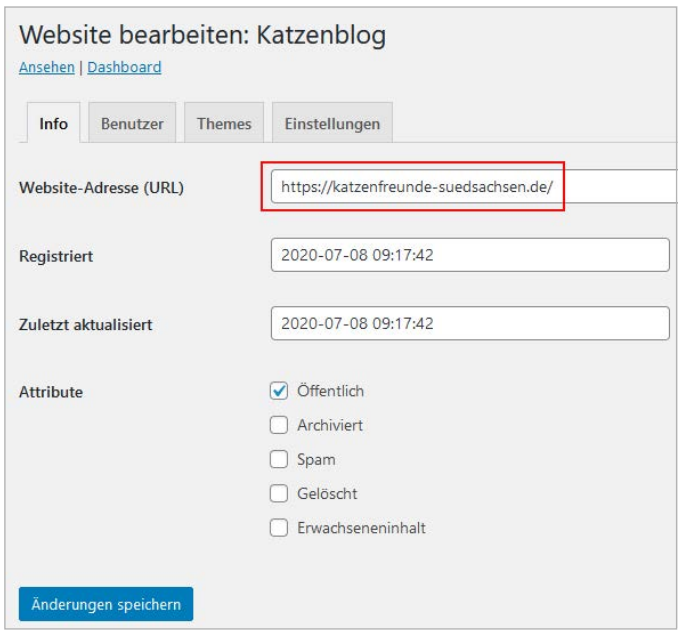
Sobald das Netzwerk aktiviert ist, erstellst Du die einzelnen Websites: **Meine Websites > Netzwerkverwaltung > Websites > Neu hinzufügen**. Als »Website-Adresse (URL)« kannst Du erstmal irgendwas eintragen – das wird eh gleich wieder geändert (Schritt 3), weil Du ja kein Unterverzeichnis, sondern eine andere Domain nutzen willst.



Beim Erstellen neuer Websites kannst Du als Adresse zunächst nur ein Verzeichnis angeben.

Schritt 3: Website-Adressen eingeben

Hast Du die Websites erstellt, musst Du nun die richtigen Adressen eingeben. Dafür gehst Du in die Website-Übersicht (**Websites > Alle Websites**). Bewegst Du den Mauszeiger über einen Eintrag, erscheint der jeweilige **Bearbeiten**-Link. Hier kannst Du eine beliebige **›STRATO Domain** aus Deinem **›Hosting-Paket** eingeben.



Tipp: Falls Du Dich nicht mehr einloggen kannst, liegt das wahrscheinlich an veralteten Cookies. Schließe in dem Fall einfach den Browser, lösche den Cache und versuche es erneut.

Fertig!

Wenn alles erledigt ist, kannst Du Dich mit Deinen »Super-Admin«-Zugangsdaten (für das gesamte Netzwerk) in die Backends der einzelnen Websites einloggen. Wenn Du im Backend einer Website einen Benutzer anlegst, hat dieser auch nur dort Zugriff. Viel Spaß mit Multisite!

8.3 WordPress mehrsprachig machen

Es gibt gute Gründe für eine mehrsprachige Website: Deine Leser kommen aus verschiedenen Ländern, Du hast eine Affinität zu einer Sprache oder willst Deine Reichweite vergrößern. Technisch ist ein mehrsprachiges Blog gar nicht kompliziert – auch wenn WordPress diese Funktion (noch) nicht von Haus aus mitbringt. Dank WordPress Multisite ist das trotzdem kein Problem.

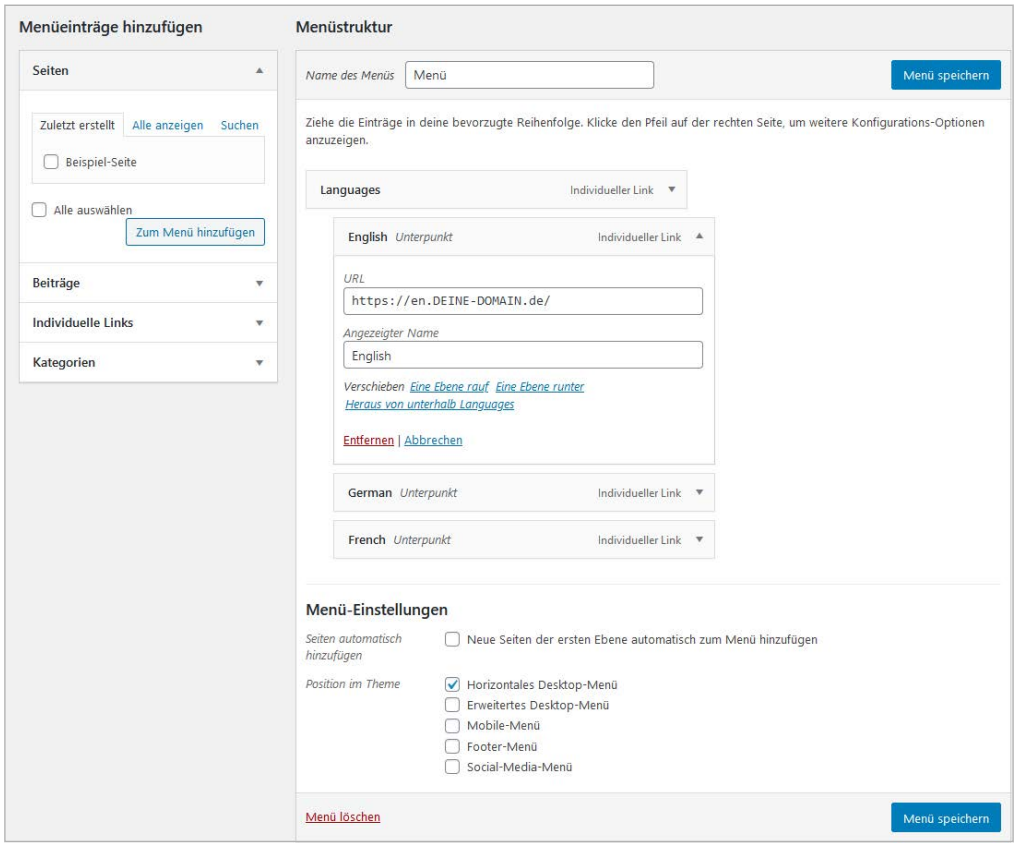
Mit Multisite kannst Du für jede Sprache ein separates Blog innerhalb einer WordPress-Installation (Netzwerk) erstellen. Das klingt komplizierter, als es ist. Hier erfährst Du Schritt für Schritt, wie Du dabei vorgehst:

1. **›Installiere WordPress manuell**,

2. **›Aktiviere WordPress Multisite** und lege für jede Sprache eine Subdomain und eine Website an.

Wichtig: Wie zuvor schon erwähnt, benötigst Du ein Wildcard-Zertifikat, um neben der Hauptseite auch die Subdomains per SSL zu verschlüsseln.

3. Anschließend solltest Du die Websites miteinander verbinden. Dazu kannst Du auf jeder Website ein Dropdown-Menü erstellen, in dem Du auf die jeweils anderen Websites verlinkst: Trage in den Menü-Einstellungen (**Design > Menüs**) die Adressen der Websites als individuelle Links ein und füge sie zum Menü hinzu. Als oberen Menüpunkt für die Sprachauswahl (zum Beispiel »Languages«) gibst Du als Adresse irgendwas an. Nur so kann WordPress den Eintrag speichern. Anschließend kannst Du den Inhalt dieses URL-Felds löschen, um die Verlinkung zu entfernen. Das Gleiche machst Du bei den anderen Websites, wobei die jeweilige Website selbst natürlich nicht verlinkt werden muss.



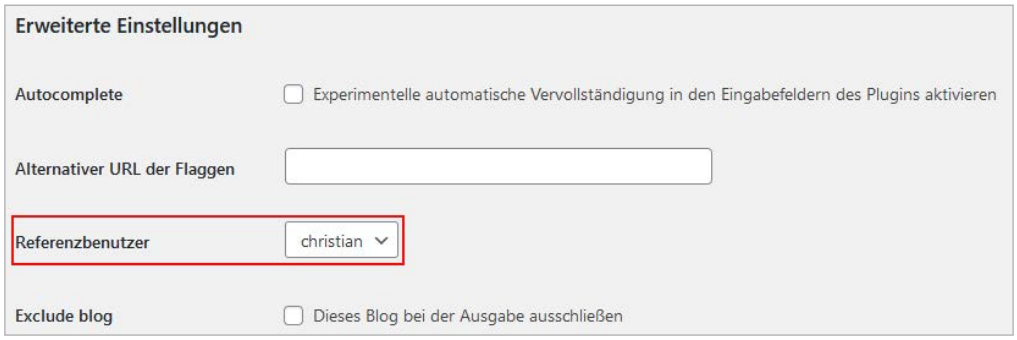
Mit Drag & Drop rückst Du die Unterpunkte ein, um ein Dropdown-Menü zu erstellen.

4. Als letzten Schritt kannst Du optional Multisite Language Switcher installieren. Das Plugin ist enorm hilfreich bei der Verwaltung der Übersetzungen. So zeigen Dir Flaggen-Icons in der Beitrags- bzw. Seitenübersicht jeder Website des Netzwerks an, ob Übersetzungen fehlen.

   Voraussetzung dafür ist, dass die jeweiligen Übersetzungen miteinander verknüpft sind:

   a) Zu diesem Zweck stellst Du zunächst für jede Website die jeweilige Sprache ein (**Einstellungen > Allgemein > Sprache der Website**).

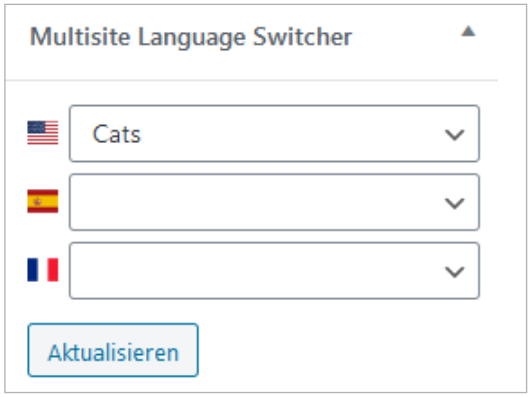
   b) Danach konfigurierst Du ebenfalls in jeder Website separat das Plugin (**Einstellungen > Multisite Language Switcher**). Achte hier auf den **Referenzbenutzer**: Dieser muss mit dem der anderen Websites identisch sein.



Den Referenzbenutzer kannst Du in den Plugin-Einstellungen festlegen.

Lege ansonsten über **Benutzer > Neu hinzufügen** einen neuen Benutzer an oder übernimm einen bestehenden, mit dem Du die zu übersetzenden Beiträge und Seiten erstellst.

c) Wenn Du Dir nun einen Beitrag oder eine Seite im Editor anschaust, findest Du unten rechts den Bereich **Multisite Language Switcher**. Hier musst Du nur noch die entsprechenden Inhalte der anderen Blogs als Übersetzungen festlegen.



Dem Beitrag »Katzen« weisen wir den Beitrag »Cats« unserer englischen (US) Website zu.

In der Beitrags-Übersicht sieht das dann wie folgt aus. Über die Icons gelangst Du direkt in den Editor des entsprechenden Inhalts:

Beitrags-Übersicht: Für den Beitrag »Katzen« gibt es in diesem Beispiel nur eine englische Übersetzung.

Schlusswort

Bist Du startklar? Steckst Du vielleicht schon mitten in Deinem WordPress-Projekt? Dann hoffen wir, dass unser E-Book »WordPress für Fortgeschrittene« für Dich ein guter Ratgeber ist.

Übrigens: Über neue WordPress-Features und Trends informieren wir Dich in unserem **›STRATO Blog**. Schau doch mal vorbei und hol Dir Anregungen für Deinen eigenen Blog.

Schließlich fragen wir uns: Was ist aus Deinem Projekt geworden? Wir sind sehr gespannt auf Dein Feedback: Besuche uns auf unserem **›Blog**, auf **›Facebook** oder **›Twitter**.

Frohes Bloggen und viele Grüße

Dein STRATO Team