david JARDIN
elisa FOLTYN

Aktuell zu Version 3.7

Joomla! 3

PROFESSIONELLE WEBENTWICKLUNG

2. Auflage

HANSER

Jardin/Foltyn

Joomla! 3

Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter

www.hanser-fachbuch.de/newsletter

Hanser Update ist der IT-Blog des Hanser Verlags mit Beiträgen und Praxistipps von unseren Autoren rund um die Themen Online Marketing, Webentwicklung, Programmierung, Softwareentwicklung sowie IT- und Projektmanagement. Lesen Sie mit und abonnieren Sie unsere News unter

www.hanser-fachbuch.de/update

David Jardin, Elisa Foltyn

Joomla! 3

Professionelle Webentwicklung.

Aktuell zu Version 3.7

HANSER

Die Autoren:

David Jardin, Köln, d.jardin@djumla.de

Elisa Foltyn, Nürnberg, book@coolcat-creations.com

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autoren und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autoren und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Dieses Werk ist urheberrechtlich geschützt.
Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2017 Carl Hanser Verlag München, www.hanser-fachbuch.de
Lektorat: Brigitte Bauer-Schiewek
Copy editing: Petra Kienle, Fürstenfeldbruck
Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de
Umschlagrealisation: Stephan Rönigk
Gesamtherstellung: Kösel, Krugzell
Printed in Germany

Print-ISBN: 978-3-446-44015-9
E-Book-ISBN: 978-3-446-44088-3

Inhalt

1 Einleitung

Liebe Leserin, lieber Leser,

man kann wohl guten Gewissens behaupten, dass sich der OpenSource-CMS-Markt aktuell im Wandel befindet. Wordpress vergrößert unaufhörlich seinen Marktanteil, Software-As-A-Service-Lösungen wie *Wix.com* oder Jimdo buhlen um die Gunst der Anwender und beinahe jeden Tag wird irgendwo ein weiteres, kleines und schlankes CMS veröffentlicht, das innovativer und besser sein möchte als alle seine Vorgänger zusammen.

Nicht wenige Kollegen aus der Web-Szene prophezeien angesichts dieser Veränderungen das baldige Verschwinden von Joomla, da es in der heutigen Zeit keinen sinnvollen Einsatzzweck mehr habe und man am besten schleunigst zu einem der anderen, vermeintlich besseren Systeme wechseln sollte.

Wir sind überzeugt, dass die Kollegen mit dieser Einschätzung kaum falscher liegen könnten. Auf dem CMS-Markt finden sich mit Wordpress, den genannten SAAS-Lösungen und diversen kleineren CMS zahlreiche Systeme die sich für die Realisierung kleiner bis mittelgroßer Seiten anbieten. Der Enterprise-Markt wird durch kommerzielle Systeme sowie TYPO3 und Drupal dominiert – aber der Markt „dazwischen" ist weder mit der einen, noch mit der anderen Kategorie von CMS sinnvoll bedienbar.

Genau hier liegt die große Stärke von Joomla – es ist der ideale Mix aus leicht erlernbarem Baukastensystem und technisch ausgereiftem Framework zur Realisierung komplexer Anforderungen. Es verfügt über ein reichhaltiges Portfolio an fertigen Erweiterungen, kann aber genauso gut durch eigene Entwicklungen ergänzt werden. Diese Fähigkeiten machen es zur eierlegenden Wollmilchsau der CMS-Industrie und somit auch in Zukunft zum Werkzeug der Wahl für Dienstleister und ambitionierte Hobbynutzer.

Mit diesem Buch möchten wir genau dieser Zielgruppe eine Hilfe an die Hand geben, die ihr beim Einstieg in Joomla hilft, ohne sie nach einer Grundeinführung im Regen stehen zu lassen. Dieses Buch basiert auf über zehn Jahren Erfahrung als Joomla-Dienstleister und fasst all die kleinen Tipps, Tricks und Workflows zusammen, die ein professionelles Arbeiten ausmachen!

Zusatzmaterial, Links und besondere Angebote, die wir Ihnen gerne zu diesem Buch zur Verfügung stellen wollen, finden Sie auf der Webseite des Hanser-Verlags unter: www.hanser-fachbuch.de/joomla3

Wir glauben an eine großartige Zukunft für Joomla und sind uns sicher, dass Sie es am Ende dieses Buchs auch tun werden!

Viel Freude bei der Lektüre wünschen Ihnen

David Jardin und Elisa Foltyn

Alle Codebeispiele aus dem Buch finden Sie auch im Internet unter
http://buch.djumla.de/3.7

2 Über Joomla!

2.1 Content-Management-Systeme

Am Anfang dieses Buchs sollten wir uns zunächst einmal mit folgender Frage beschäftigen: Was ist eigentlich ein Content-Management-System (kurz CMS)? Die deutsche Übersetzung „Inhaltsverwaltungssystem“ hilft uns ein wenig weiter: Ein CMS ist eine Software zur Erstellung, Bearbeitung und Verwaltung von Informationen, die aus simplen Texten, aber auch aus komplexen Multimediaelementen (Bilder, Videos, Dokumente etc.) bestehen können. Dabei ist wichtig, dass die entsprechenden Inhalte im Regelfall ohne Programmierkenntnisse eingepflegt werden können.

Obwohl auch sog. Document-Management-Systeme (z. B. Alfresco) oder Offline-CMS wie Jekyll im weitesten Sinne zu den Content-Management-Systemen gehören, bezieht sich der Begriff des CMS i. d. R. auf Web-Content-Management-Systeme, die *ausschließlich* als Webanwendungen arbeiten, also über den Webbrowser administriert werden.

Diese WCMS erlauben es mehreren Benutzern, gemeinschaftlich an den hinterlegten Informationen zu arbeiten, und sind medienneutral in ihrer Ausgabe. Medienneutral bedeutet dabei, dass die hinterlegten Informationen unabhängig vom Ausgabeformat (HTML, PDF, XLS) bzw. der Gestaltung der Ausgabe (einfaches Wechseln von Designs) hinterlegt sind und so aus einem „Inhaltspool“ verschiedene Arten von Dokumenten erzeugt werden können. Dies wird dadurch ermöglicht, dass die Inhalte erst beim Aufruf durch den Nutzer *dynamisch* in ihr finales Ausgabeformat gebracht werden. Das unterscheidet Web-CMS vom klassischen Vorgehen mit HTML-Editor und FTP-Programm, bei dem die einzelnen Inhalte *statisch*, also schon in ihrer endgültigen Form, auf dem Server hinterlegt sind.

Zu den bekanntesten Open-Source-CMS gehören das Blogsystem Wordpress, die Enterprise-Systeme Drupal und Typo3 sowie Joomla!.

2.2 Geschichte

Die Wurzeln von Joomla! liegen im CMS Mambo, das seit der Jahrtausendwende vom australischen Unternehmen Miro entwickelt und im Jahr 2002 als Open-Source-Software veröffentlicht wurde. Mambo entwickelte sich schnell zu einem sehr beliebten System und wurde so z. B. im Jahr 2004 von der Zeitschrift *Linux User and Developer* als „Best Linux or Open Source Software" ausgezeichnet. Im August 2005 entschied sich Miro dazu, die Mambo Foundation, einen gemeinnützigen Verein, zu gründen, um das Mambo-Projekt von Miro zu lösen und so zu gewährleisten, dass die Weiterentwicklung unabhängig vom Schicksal der Firma erfolgen kann.

Einige Tage später kam es jedoch zum Bruch zwischen dem aus Freiwilligen bestehenden Entwicklerteam und Miro, woraufhin das gesamte Entwicklerteam das Mambo-Projekt verließ, um sich unter dem Namen „Open Source Matters" neu zu gruppieren. Als Grund für diesen Schritt gab das Entwicklerteam an, dass die Mambo Foundation ohne Beteiligung der Community gegründet worden sei und Miro weiterhin eine starke Kontrollfunktion ausübe, die mit einem Open-Source-Projekt nicht vereinbar wäre – so ließ sich z. B. der Geschäftsführer von Miro zum Vorsitzenden der Foundation wählen.

Das Entwicklerteam entschloss sich daraufhin, einen eigenen Ableger von Mambo auf den Markt zu bringen, der im September 2005 unter dem Namen Joomla! in der Version 1.0 erschien. Joomla! leitet sich vom Suaheli-Wort „Jumla" ab, das in der Übersetzung „Alle zusammen" bedeutet. Joomla! 1.0 war zu diesem Zeitpunkt im Wesentlichen nur eine leicht fehlerbereinigte Version von Mambo 4.5.2.3, zog jedoch aufgrund des Wechsels des gesamten Entwicklerteams große Teile der Mambo-Community mit sich.

Nach der Veröffentlichung von Joomla! 1.0 und der Stabilisierung des Projekts entschloss sich das Entwicklerteam, den alten Code, der teilweise noch aus dem Jahr 2000 stammte, über Bord zu werfen und eine von Grund auf neu geschriebene Joomla!-Version 1.5 zu erstellen, die im Januar 2008 erschien und dem Projekt nochmals einen enormen Aufwind gab. In den aktuellen Versionen findet sich dadurch kein Code des Vorgängers Mambo mehr.

2.3 Organisation

Das Joomla!-Projekt ist in den letzten Jahren massiv gewachsen und musste dabei seine Strukturen mehrfach an die veränderten Anforderungen anpassen. Die aktuelle Organisationsstruktur befindet sich zum Zeitpunkt des Erscheinens dieses Buchs in einem Übergangsprozess, ist also noch nicht vollständig umgesetzt worden.

Die Grundstruktur des Projekts entspricht auf den ersten Blick der eines klassischen Unternehmens (siehe Bild 2.1).

Bild 2.1 Neue Organisationstruktur des Joomla!-Projekts

Basis des Projekts ist die juristische Person hinter Joomla!, nämlich die bereits erwähnte not-for-profit-Organisation „OpenSourceMatters Inc“ mit Sitz in New York.

OpenSourceMatters hat einen Vorstand, bestehend aus den Sonderrollen „Präsident/in“, „Vize-Präsident/in“, „Generalsekretär/in“ und „Schatzmeister/in“ sowie den Abteilungsleiter/innen der sieben Abteilungen, im Projektjargon als „Department Coordinator“ bezeichnet.

Derzeit sind die folgenden Abteilungen im Projekt vorgesehen:

- Production: zuständig für die technische Entwicklung des eigenen Kernprodukts „Joomla! CMS“.
- Legal: verteidigt die rechtlichen Interessen des Gesamtprojekts, insbesondere in Bezug auf die Marke „Joomla!“.
- Marketing & Communication: koordiniert die interne und externe Kommunikation des Projekts und versucht die Verbreitung von Joomla! zu erhöhen.
- Events: zuständig für alle Fragen rund um Veranstaltungen und Meetups.
- Operations: kümmert sich um den Betrieb der Infrastruktur, den das Projekt benötigt, wie zum Beispiel die Website *joomla.org*.
- Programs: betreut die verschiedenen Programme und Initiativen, an denen Joomla! beteiligt ist. Hierzu gehört zum Beispiel das „Google Summer of Code“-Programm oder das geplante Zertifizierungs-Programm für Joomla!-Administratoren und Dienstleister.
- Local Communities: repräsentiert die lokalen Communities wie Usergroups oder nationale Vereine.

Jedes Department kann wiederum aus beliebig vielen Teams bestehen, die für jeweils einen bestimmten Aspekt zuständig sind. Im Department „Production“ könnte es hier z. B. ein Team für die Übersetzung von Joomla! in andere Sprachen geben, ein Team für die Entwicklung von Joomla! 4.x sowie ein Team für die Dokumentation. Die Anzahl der Teams unterliegt dabei keiner Begrenzung, sondern kann frei gewählt werden.

Auf der Ebene der Teams angekommen besteht jedes Team aus den Positionen „Teamleiter/in“ und „stellv. Teamleiter/in“ sowie einer beliebigen Anzahl von Teammitgliedern. Die Aufnahme in ein Team erfolgt über ein Bewerbungsverfahren, das die Teams individuell gestalten können. Ein Team wiederum wählt dann den Teamleiter sowie seinen Stellvertreter, die Teamleiter eines Departments wählen den Department Coordinator und alle Department Coordinators wählen die passenden Personen für die vier genannten Sonderposten auf Vorstandsebene.

Die Projektstruktur verfügt somit über einige bemerkenswerte Merkmale:

Absolut alle Mitarbeiter im Projekt, ganz egal auf welcher Ebene, arbeiten unbezahlt und ehrenamtlich.

Es gibt nicht „die Firma“ hinter dem Projekt, die die Entwicklung steuert, sondern die Richtungsfindung und Entwicklung erfolgt im Rahmen von demokratischen Prozessen.

Es gibt klare Prozesse für Abstimmungen, Teamgründungen und -auflösungen sowie die Aufnahme neuer Mitarbeiter.

An dieser Stelle sollte jedoch nicht verschwiegen werden, dass die neue Struktur in der Community nicht unumstritten ist. Die Abstimmung über den Wechsel zum neuen Aufbau ging denkbar knapp aus und der derzeit stattfindende Wechsel von der alten auf die neue Struktur ist massiv hinter dem aufgestellten Zeitplan. Kritiker äußern dabei vor allem die folgenden Kritikpunkte:

Die neue Struktur ist zu bürokratisch. Langatmige Prozesse verlangsamen die Entscheidungsfindung und nehmen allen Beteiligten die nötige Flexibilität.

In der neuen Struktur konzentriert sich zu viel „Macht“ auf einige wenige Menschen, nämlich den Vorstand von OpenSourceMatters. Ein System der gegenseitigen Prüfung von mehreren gleichberechtigten Instanzen, wie es in der alten Struktur der Fall war, fehlt.

Der Wechsel zur neuen Struktur würde das Projekt über Monate beschäftigen und lähmen.

Zum jetzigen Zeitpunkt (Februar 2017) ist noch nicht absehbar, wie die neue Struktur sich auf das Projekt auswirkt – man darf daher gespannt sein.

Als Mitglied des Joomla! Community Leadership Teams war ich, der Autor dieses Kapitels, direkt an der Abstimmung über die Adaption der neuen Struktur beteiligt und habe dabei gegen deren Einführung gestimmt. Ich habe mich bemüht, die Struktur dennoch so neutral wie möglich darzustellen und hoffe, dass mir das gelungen ist. Nichtsdestotrotz sind Sie herzlich eingeladen, die offizielle Beschreibung[1] des Strukturentwurfs zu lesen und sich ein eigenes Bild zu machen.

[1] *https://docs.google.com/document/d/1gsUK0kePsBg6xiaUVdN6oExZ0hKEjBxH6bRVAeZy-IE/edit?usp=sharing*

2.4 Release-Strategie

Mit der Veröffentlichung von Joomla! 3.2 hat das Joomla!-Projekt einen Wechsel der Release-Strategie beschlossen. Das System des sog. Time-Based-Releasecycle, das zum Release von Joomla! 1.6 im Jahr 2011 eingeführt wurde und auf der Idee von fest terminierten Veröffentlichungen und Langzeit- und Kurzzeitsupport-Versionen basierte, wurde aufgegeben und stattdessen eine Strategie auf Basis der beiden Grundsätze *„schlanke, schnelle Releases"* und *„semantische Versionierung"* eingeführt.

Schlanke, schnelle Releases bedeuten dabei, dass neue Joomla!-Versionen in relativ kurzen zeitlichen Abständen (im Idealfall ca. ein Release pro Quartal) veröffentlicht werden sollen und die jeweiligen Releases dabei jeweils nur relativ kleine überschaubare Sets an neuen Funktionen mitbringen sollen. Das gegenteilige Modell wäre z. B. die Veröffentlichung von nur einem Update pro Jahr, das dann aber wesentlich umfangreicher ist und eine Vielzahl von neuen Funktionen mitbringt. Dieser Ansatz erlaubt den Joomla!-Entwicklern, sehr schnell auf neue Anforderungen der Webwelt zu reagieren und führt zudem zu schnellen Erfolgserlebnissen für Entwickler, die eine Funktion zu Joomla! beisteuern.

Der zweite Grundsatz, die sogenannte semantische Versionierung, gibt vor, welche Art von Änderung sich auf welche Stelle der Versionsnummer auswirkt. Eine dreistellige Versionsnummer X. Y. Z lässt sich dabei in die folgenden Komponenten aufteilen:

1. X ist die sog. Majorversion. Eine Änderung dieser Ziffer ist notwendig, wenn das Joomla!-Projekt eine Änderung einbaut, die einen Bruch der Rückwärtskompatibilität zur Folge hat. Mit anderen Worten: Ändert sich die erste Stelle der Versionsnummer, müssen Entwickler ihre Erweiterungen an die neue Version anpassen, da diese ansonsten nicht mehr lauffähig sind. Somit ist der Wechsel von der einen zur anderen Majorversion ein etwas komplexerer Prozess, der im Joomla!-Jargon als Migration bezeichnet wird und in Kapitel 22 genauer beschrieben ist.
2. Y ist die sog. Minorversion. Hier ist eine Änderung immer dann notwendig, wenn eine neue Funktion zu Joomla! hinzugefügt wird. Diese Minorversionen erscheinen im oben bereits beschriebenen Quartalsrhythmus. Ein Update auf eine neue Minorversion ist sehr simpel und in der Regel mit einem einfachen Mausklick durchführbar.
3. Z ist die sog. Patchversion. Diese kleinste Art von Update enthält ausschließlich Fehlerbehebungen (auch als Patches bezeichnet) und kann ebenfalls bedenkenlos per Mausklick direkt in der Administration von Joomla! eingespielt werden.

Um den Nutzern von Joomla! eine gewisse langfristige Planungssicherheit zu geben, gibt das Joomla!-Projekt feste Mindestzeiträume an, in denen eine bestimmte Version noch mit Sicherheitsupdates und Fehlerbehebungen versorgt wird. Grundregel ist dabei, dass die jeweils letzte Minor-Version eines Major-Zweigs für zwei Jahre unterstützt wird.

Ein kleines Beispiel, um diesen etwas abstrakten Satz mit konkreten Inhalten zu füllen:

Wir nehmen einmal an, dass am 01. 01. 2018 eine fiktive Joomla!-Version 4.0 erscheint. Im weiteren Entwicklungsverlauf erscheinen für diesen neuen Versionszweig 4.x nun mehrere Minor-Releases, die neue Funktionen nachrüsten. Die letzte Minor-Version, in unserem fiktiven Fall wäre das zum Beispiel 4.9.0, erscheint dabei am 01. 03. 2020. Der Release-Termin dieser letzten Minor-Version wäre nun ausschlaggebend für das Ende des Supportzeitraums

von zwei Jahren, womit der Support für die Joomla!-Version 4.x am 28.02.2022 enden würde. Im Rahmen dieses Supportzeitraums würde es weiterhin Fehlerbehebungen und Sicherheitsupdates geben, die sich dann auf die letzte Ziffer der Versionsnummer (z.B. als 4.9.1, 4.9.2 etc.) auswirken würden.

Joomla! bietet Entwicklern und Nutzern also eine langfristige Planungssicherheit und eignet sich daher perfekt für Projekte, die über einen längeren Zeitraum unterstützt werden müssen.

3 Einrichten der Arbeitsumgebung

Nachdem wir uns eingehend mit der Joomla!-Geschichte beschäftigt haben, machen wir uns nun an die Arbeit. Bevor wir jedoch mit der Installation unserer Joomla!-Umgebung loslegen können, gibt es noch einige äußerst wichtige Schritte zu erledigen, in denen wir unsere lokale Arbeitsumgebung, gewissermaßen unsere „Werkstatt", einrichten werden. Dabei brauchen wir vor allem vier Programme bei unserer weiteren Arbeit:

- Einen *lokalen Webserver* mit PHP und MySQL, um neue Joomla!-Seiten lokal auf dem eigenen Rechner entwickeln zu können. Natürlich kann man dies auch auf einem angemieteten Webspace tun, jedoch dauern viele Handgriffe durch den Umweg über FTP deutlich länger.
- Einen passenden *Texteditor* zur Editierung von HTML, CSS, JavaScript und PHP sowie eine entsprechende *Entwicklungsumgebung* zur Programmierung eigener Joomla!-Erweiterungen.
- Einen *Webbrowser* inklusive der benötigten Erweiterungen für die Webentwicklung.
- Einen *FTP-Client* zum Transfer von Daten zum Webspace.

Wenn Sie die entsprechenden Tools Ihrer Wahl bereits gefunden haben, möchte ich Ihnen trotzdem nahelegen, das Kapitel zumindest kurz zu überfliegen, denn vermutlich gibt es auch für Sie noch die eine oder andere Kleinigkeit, die Ihnen die Arbeit erleichtern kann.

3.1 Lokaler Webserver

Joomla! ist kein ausführbares Programm im engeren Sinn, weshalb es, anders als bei vielen anderen Programmen, keine obligatorische *Joomla.exe* gibt, welche die Software startet. Stattdessen ist für die Ausführung eine weitere Software notwendig, welche die über den Browser erstellten Nutzeranfragen verarbeitet, Joomla! ausführt und das Ergebnis an den Nutzer zurückgibt. Diese Software ist ein sogenannter *Webserver*, deren bekanntester Vertreter der *Apache*-Server ist. Der Webserver muss die Ausführung von *PHP*-Skripten unterstützen, um Joomla! ausführen zu können, und gleichzeitig muss eine *MySQL*- oder *MariaDB*-Datenbank installiert sein, um die Daten unserer Seite speichern zu können. Diese Kombination aus **A**pache, **M**ySQL bzw. *M*ariaDB und **P**HP wird auf **L**inux-Systemen als *LAMP* bezeichnet,

woraus sich die Namen für die anderen Betriebssysteme (*MAMP* unter OS X, *XAMPP* unter Windows) ableiten.

Auf den folgenden Seiten möchte ich für jedes der genannten Systeme die Installation des benötigten *Webservers* beschreiben, wobei sich, je nach Systemumgebung, einige Detailschritte unterscheiden können.

HINWEIS: Die im Folgenden beschriebene Webserverkonfiguration eignet sich ausschließlich zum Betrieb von lokalen Testumgebungen und sollte unter keinen Umständen als produktiver Webserver für eine öffentlich erreichbare Website genutzt werden!

3.1.1 Windows

Unter Windows starten wir mit der Installation des Webserver-Pakets *XAMPP*, indem wir das Installationspaket von der Homepage der Entwickler (*www.apachefriends.org*) herunterladen.

Bild 3.1 Download des *XAMPP*-Pakets von der Entwickler-Homepage

Nach dem Herunterladen starten Sie die Installation mit einem Doppelklick auf den Installer. Folgen Sie den Anweisungen des Installationsprogramms und belassen Sie dabei den Pfad zum Zielverzeichnis bei der Standardvorgabe *c:\xampp*.

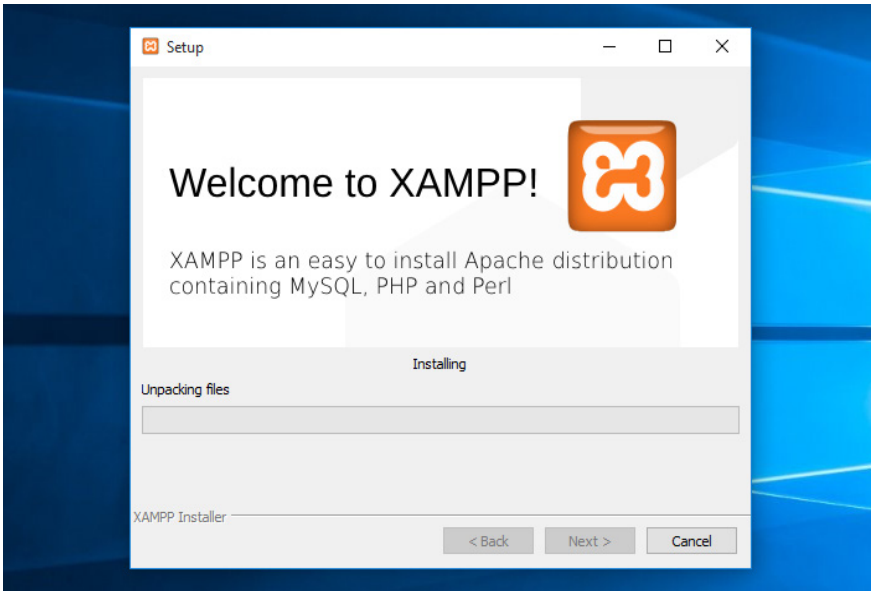


Bild 3.2 Installationsprozess von *XAMPP*

Nach der Installation öffnet sich das *XAMPP Control Panel*, in dem wir nun per Klick den Apache- und den Datenbank-Server starten. Sollten an dieser Stelle Warnmeldungen von UAC oder installierten Firewall-Tools auftreten, so müssen Sie den beiden Diensten den Netzwerkzugriff erlauben.

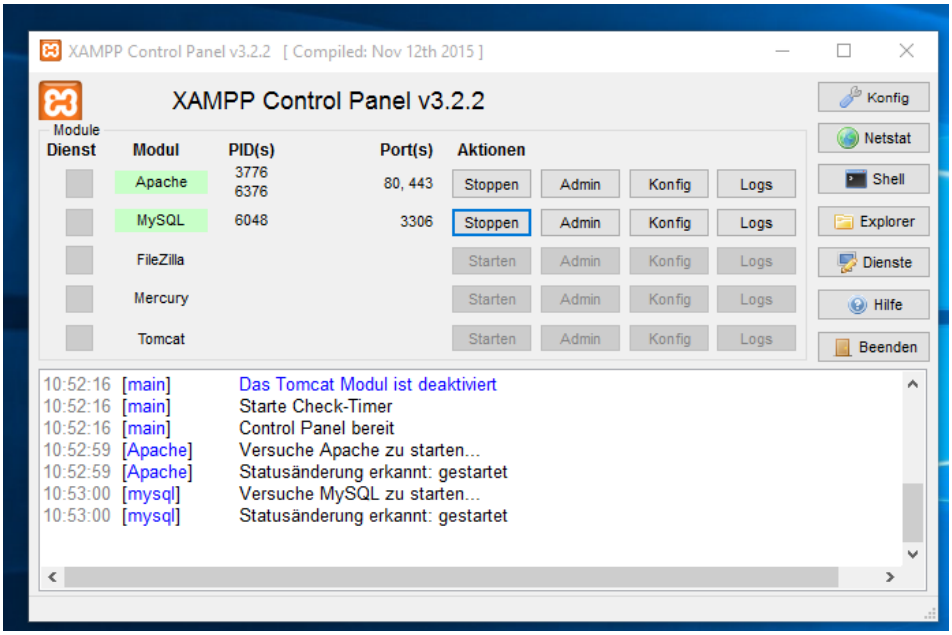


Bild 3.3 *XAMPP Control Panel*

Anschließend können wir unseren Browser aufrufen und durch die Eingabe der URL *http://localhost/dashboard/* die Startseite des gerade installierten Webservers aufrufen. Nun können wir mit der Joomla!-Installation fortfahren, da keine weiteren Anpassungen am Webserver nötig sind, um Joomla! zu betreiben.

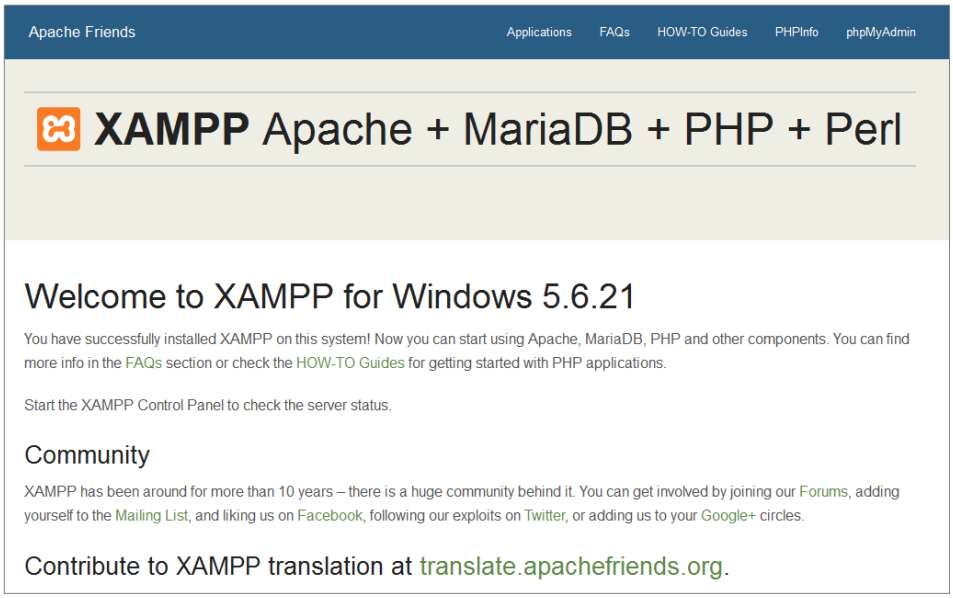


Bild 3.4 Startseite des *XAMPP*-Pakets unter Windows

3.1.2 Linux

Die Installation der Webserverumgebung LAMP (Linux, Apache, MySQL, PHP) unter Linux unterscheidet sich natürlich von Distribution zu Distribution, weshalb ich mich an dieser Stelle auf die Beschreibung der Einrichtung unter *Ubuntu Linux 16.04* beschränken möchte. *LAMP*-Installationsanleitungen für Ihre Distribution finden Sie mit ein wenig Suchmaschineneinsatz im Internet.

HINWEIS: Alternativ zur Nutzung der LAMP-Pakete in der jeweiligen Distribution ist auch die Nutzung des separaten XAMPP-Pakets unter Linux möglich – dieses wird dann jedoch selbstverständlich nicht über die Paketverwaltung des Betriebssystems aktualisiert, sodass Sie dieses Paket manuell auf dem aktuellen Stand halten müssen. Eine Anleitung zur Installation von XAMPP unter Linux finden Sie im Joomla!-Dokumentationswiki unter: *https://docs.joomla.org/Configuring_a_XAMPP_server_for_joomla_development*

Beginnen Sie die Installation, indem Sie ein *Terminal* öffnen und dort den Befehl

```
sudo apt-get install mysql-server php7.0-mysql mysql-client php7.0 libapache2-
mod-php7.0 apache2 phpmyadmin
```

ausführen. Daraufhin beginnt *Ubuntu* mit dem Herunterladen der benötigten Software und installiert diese.

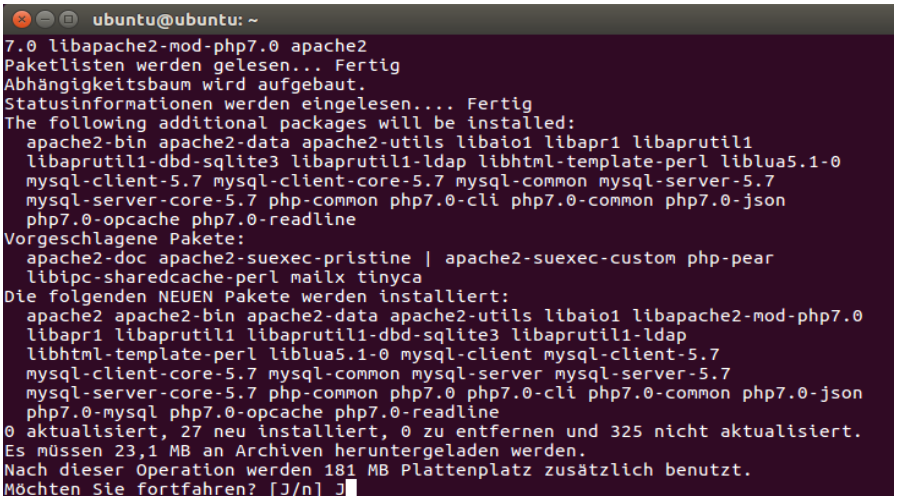


Bild 3.5 Installation des LAMP-Pakets unter Ubuntu

Im Verlauf der Installation werden Sie nun um die Eingabe eines *MySQL*-Root-Passworts gebeten, das als administratives Passwort für den *MySQL*-Server dient. Vergeben Sie hier ein Wunschpasswort und merken Sie sich dieses dauerhaft.

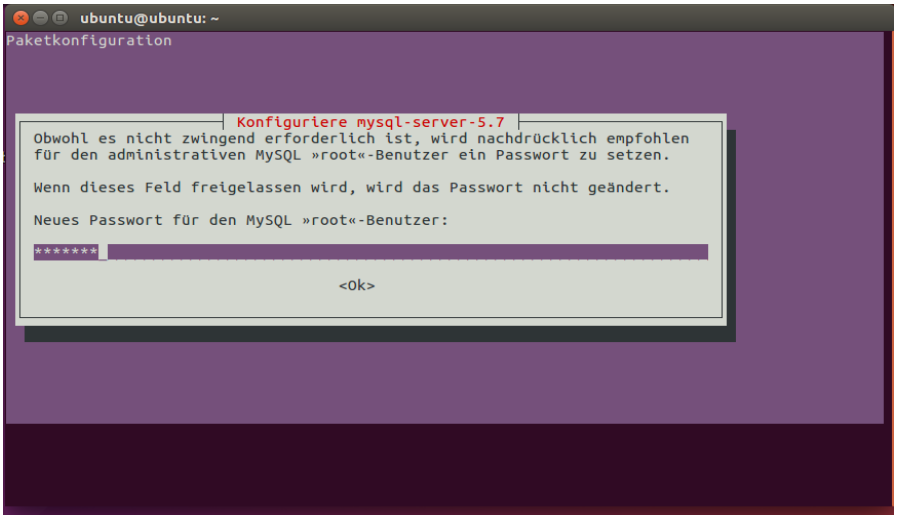


Bild 3.6 Eingabe des gewünschten MySQL-Root-Passworts

Nun ist Ihr neu installierter Webserver bereits unter *http://localhost/* erreichbar, benötigt jedoch noch einige Anpassungen, um mit unserer gewünschten Joomla!-Umgebung zu harmonieren.

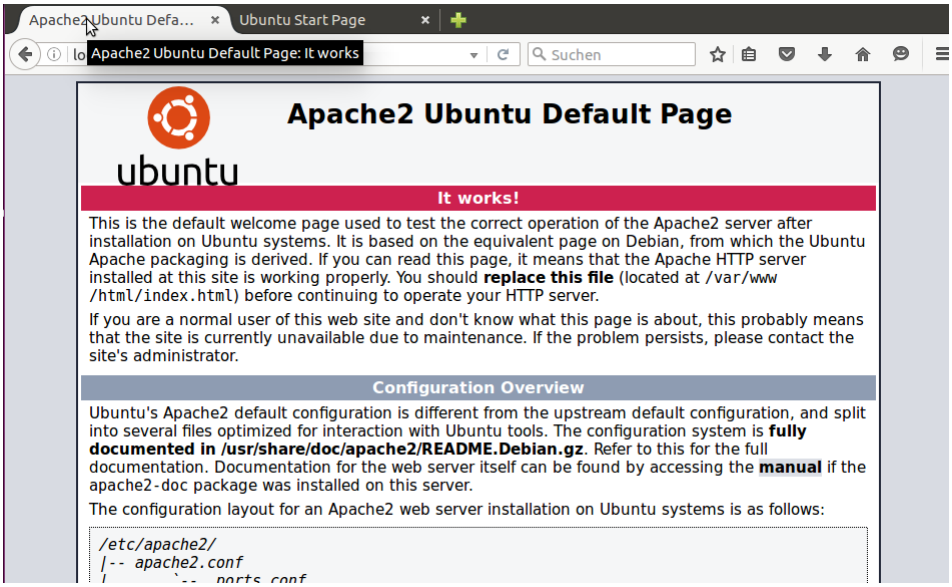


Bild 3.7 Webserver nach der Installation

Diese Anpassungen sind nötig, weil der *Webserver* unter *Ubuntu* standardmäßig mit einem eigenen Benutzernamen betrieben wird, der gleichzeitig auch Inhaber des *Docroot*-Verzeichnisses (*/var/www*) des *Webservers* ist. Deshalb ist es uns nicht möglich, mit unserem eigenen Benutzernamen Änderungen am Code der Joomla!-Installation vorzunehmen. Wir verändern also die Konfiguration des *Apache*, damit dieser stattdessen unter unserem eigenen Benutzernamen läuft, wodurch wir problemlos auf das entsprechende Docroot-Verzeichnis zugreifen können.

HINWEIS: Den *Apache*-Server mit den Rechten des eigenen Benutzernamens laufen zu lassen, bringt einige Sicherheitsrisiken mit sich, weshalb wir dieses Verfahren ausschließlich in unserer lokalen Umgebung anwenden, die keine Zugriffe von außen zulässt. Wenden Sie dieses Verfahren niemals auf Produktivsystemen an!

Um die Änderungen auszuführen, öffnen Sie die Datei */etc/apache2/envvars* mit einem Editor Ihrer Wahl (hier *nano*):

```
Sudo nano /etc/apache2/envvars
```

Und ändern dort die Zeilen

```
export APACHE_RUN_USER=www-data
export APACHE_RUN_GROUP=www-data
```

dahingehend ab, dass *www-data* durch Ihren eigenen Benutzernamen ersetzt wird:

```
export APACHE_RUN_USER=djardin
export APACHE_RUN_GROUP=djardin
```

Anschließend editieren wir noch die Datei */etc/apache2/ports.conf* und ersetzen dort den Eintrag

```
Listen 80
```

durch

```
Listen 127.0.0.1:80
```

und weisen den *Apache* dadurch an, nur lokale Verbindungen anzunehmen.

Anschließend ändern wir noch den Inhaber des Docroot-Verzeichnisses und starten den Webserver neu:

```
sudo chown USERNAME:USERNAME –R /var/www
sudo /etc/init. d/apache2 restart
```

Nun ist das System bereit für die Joomla!-Installation.

3.1.3 Mac OS X

Die Installation unseres lokalen *Webservers* unter Mac OS X kann prinzipiell über drei verschiedene Wege erfolgen:

1. Nutzung bzw. Konfiguration der ohnehin bereits vorhandenen Webserver-Komponenten
2. Nutzung des speziell für OS X geschriebenen *MAMP*-Pakets (**M**ac OS X, **A**pache, **M**ySQL, **P**HP)
3. Nutzung der OS X-Version von *XAMPP*

Ich möchte Ihnen an dieser Stelle zu Variante 2, also der Nutzung von MAMP raten, da die Konfiguration des integrierten Webservers relativ aufwendig ist und die OS X-Version von *XAMPP* leider nicht mit dem Komfort von *MAMP* mithalten kann. *MAMP* existiert in zwei verschiedenen Versionen (Standard und PRO), wobei die kostenlose Standardversion für unsere Zwecke vollkommen ausreichend ist.

Die Installation von MAMP beginnt mit dem Download des Installationspakets von der Homepage des Projekts unter *http://www.mamp.info*.

Bild 3.8 Homepage des MAMP-Projekts mit Download-Möglichkeit

Nach dem Download entpacken wir das Paket und starten die Installation durch einen Doppelklick auf die gerade entpackte Datei *MAMP_MAMP_PRO_X.pkg*. Daraufhin werden wir vom Installer durch die Installation des Webservers geführt.

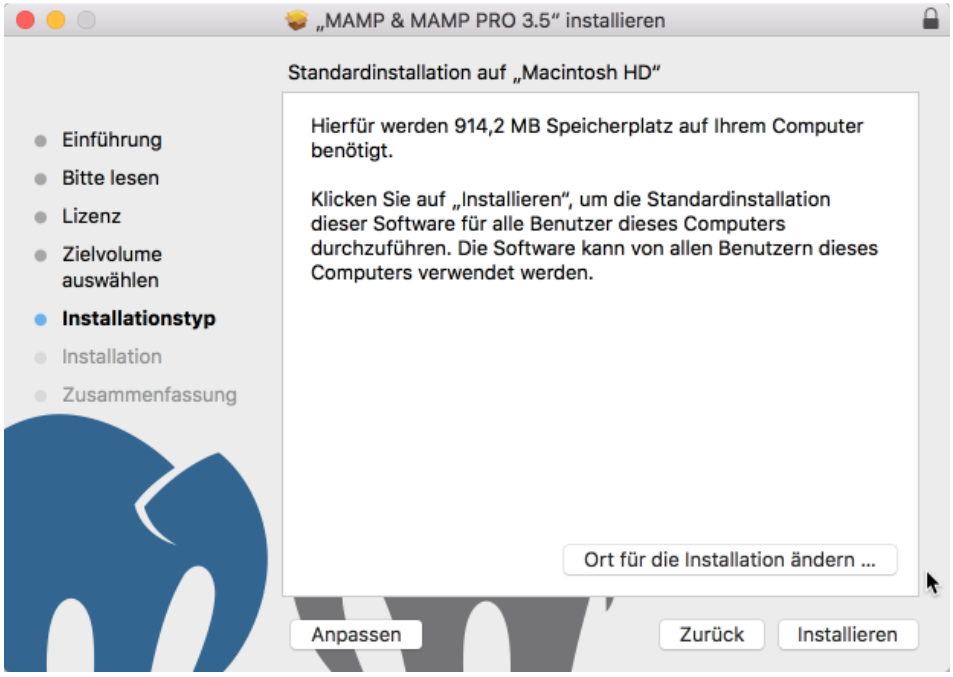


Bild 3.9 Installation des *MAMP*-Pakets

Nach der erfolgreichen Installation können wir *MAMP* und *MAMP PRO* in unserem *Programme*-Ordner finden, wobei sich die PRO-Version selbstverständlich nur nach dem Kauf der entsprechenden Lizenz nutzen lässt. Daher starten wir die Standardversion durch einen Doppelklick auf das entsprechende Icon im *Programme*-Ordner.

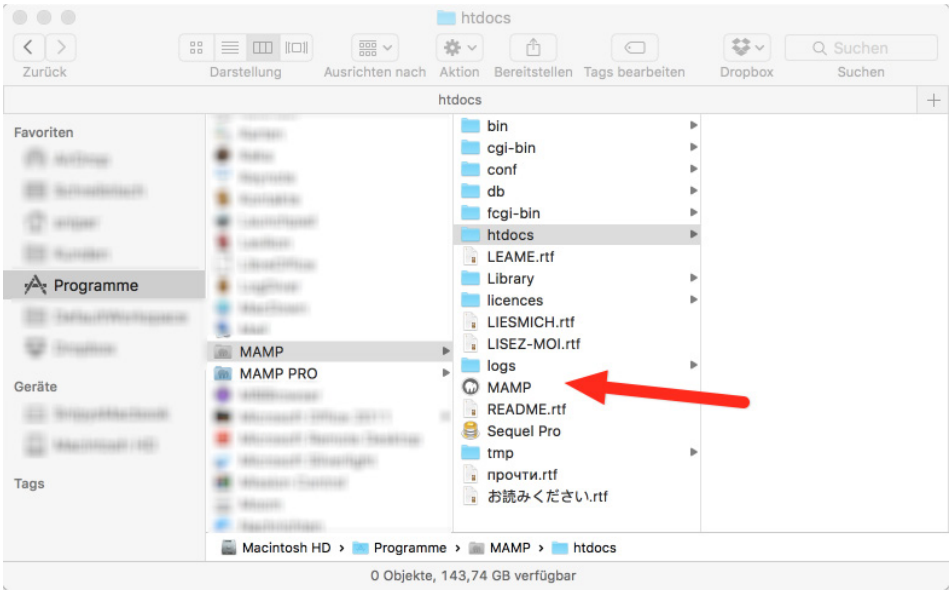


Bild 3.10 Start von MAMP mittels Verknüpfung im Programmordner

Anschließend öffnet sich das *MAMP*-eigene Kontrolltool, das uns per Mausklick das Starten und Beenden des Webservers erlaubt. Zudem können wir in den Einstellungen die zu verwendende *PHP-Version*, die *Web-* und *MySQL*-Server-Ports sowie das sog. *Document Root*, also das Hauptverzeichnis des *Webservers*, wählen. Weitere Anpassungen an *MAMP* sind zum Betrieb von Joomla! nicht nötig.

Bild 3.11 MAMP-Kontrolltool

3.1.4 Vagrant

Eine Alternative zur Installation auf dem eigenen Rechner stellt die Nutzung einer virtuellen Umgebung auf Basis des Tools *Vagrant* dar. Vagrant ist ein Werkzeug, das es dem Benutzer erlaubt, automatisch eine vorkonfigurierte virtuelle Maschine zu erzeugen. Vagrant wird in Entwicklerkreisen gerne für lokale Entwicklungsumgebungen genutzt, die auf Knopfdruck erzeugt werden können und dabei nicht vom verwendeten Betriebssystem abhängen oder erst aufwendig eingerichtet werden müssen.

Für Joomla! existiert eine speziell vorbereitete Vagrant-Umgebung, die von den Kollegen von *joomlatools.eu* vorbereitet wurde. Diese Vagrant-Box heißt schlicht Joomlatools Vagrant und bringt neben einem lokalen Webserver noch diverse vorinstallierte Debugging- und Administrationswerkzeuge mit. Eine Anleitung zur Installation finden Sie im Github Repository des Projekts unter *https://github.com/joomlatools/joomlatools-vagrant*.

Zu beachten ist dabei, dass sich durch die Nutzung der Vagrant-Umgebung einige Arbeitsabläufe verändern, weshalb die folgenden Kapitel, insbesondere die Teile, die die Administration der Seite betreffen, nur noch bedingt zutreffen würden. Ich empfehle Ihnen daher, die Installation zunächst manuell vorzunehmen und sich mit dem Thema Vagrant dann zu einem späteren Verlauf erneut zu beschäftigen.

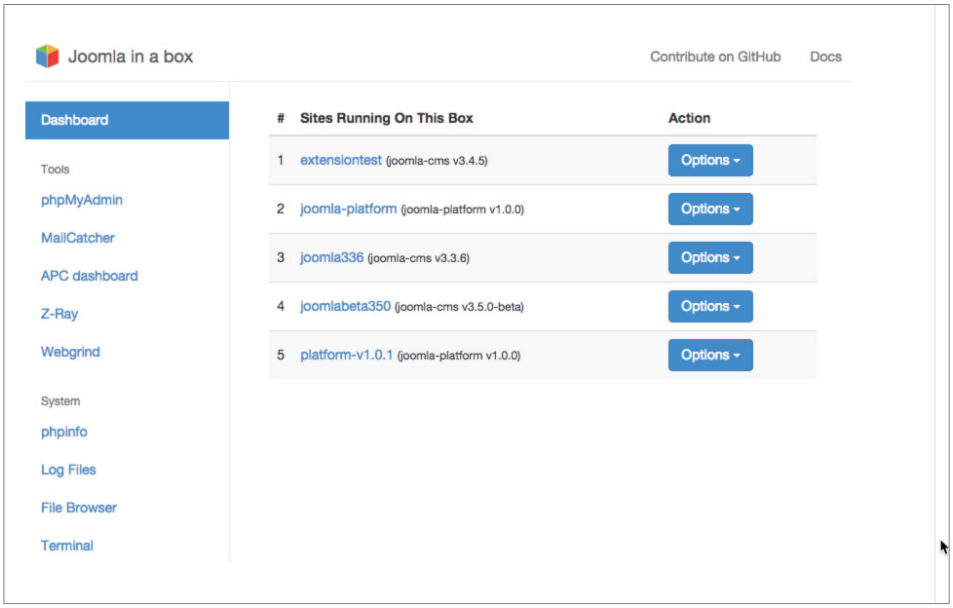


Bild 3.12 Screenshot der Joomlatools-Vagrant-Umgebung

3.2 Entwicklungstools

Als zukünftiger Webentwickler benötigen Sie neben dem nun fertig installierten Webserver natürlich auch Tools zum Editieren des HTML-, CSS- und PHP-Quellcodes. Warum mehrere Tools? Reicht nicht auch eines? Nun, prinzipiell können Sie alle benötigten Änderungen natürlich auch mit einem einfachen *Texteditor* durchführen. Gerade bei der Entwicklung komplexer Erweiterungen für Joomla! bietet sich jedoch die Nutzung einer *Entwicklungsumgebung* (IDE, Integrated Development Environment) an, die uns die Arbeit durch Funktionen wie Autovervollständigung oder Syntaxprüfung erleichtern kann. Umgekehrt lohnt es sich für kleinere Änderungen oft nicht, eine umfangreiche *IDE* zu starten, weshalb sich hier eher die Nutzung eines einfachen *Texteditors* empfiehlt.

Ich möchte Ihnen daher für jedes der gängigen drei Betriebssysteme (*Linux, Windows, OS X*) je einen *Texteditor* und eine *Entwicklungsumgebung* vorstellen, die sich bei mir im täglichen Gebrauch bewährt haben. Letzten Endes müssen Sie aber Ihren ganz persönlichen Favoriten finden, mit dem Sie am einfachsten arbeiten können.

3.2.1 Texteditor

3.2.1.1 Windows: Notepad++

Notepad++ (*notepad-plus-plus.org*) lädt dank seines sehr schnellen Codes und seines überschaubaren Funktionsumfangs binnen Sekunden und bietet eine sehr gute UTF-8-Unterstützung, was ihn zu meiner ersten Wahl für die Durchführung kleinerer Änderungen auf Windows-Systemen macht.

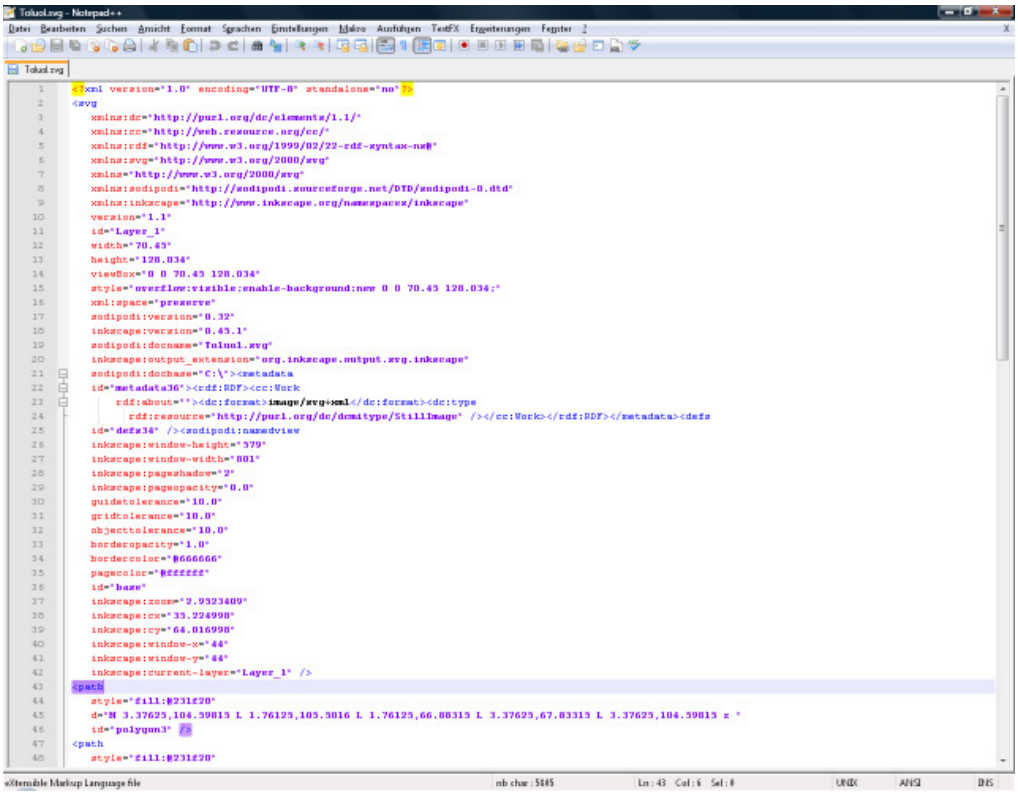


Bild 3.13 Notepad++ unter Windows

3.2.1.2 Alle Systeme: Sublime Text

Der Quasi-Standard für schnelle Änderungen ist inzwischen der Texteditor Sublime Text. Der Editor ist für alle gängigen Betriebssysteme erhältlich und glänzt neben seiner herausragenden Performance vor allem durch interessante Bedienkonzepte (wie zum Beispiel Mehrfachcursor) und durch sein flexibles Plug-in-System, womit Sublime sich an den eigenen Geschmack anpassen lässt.

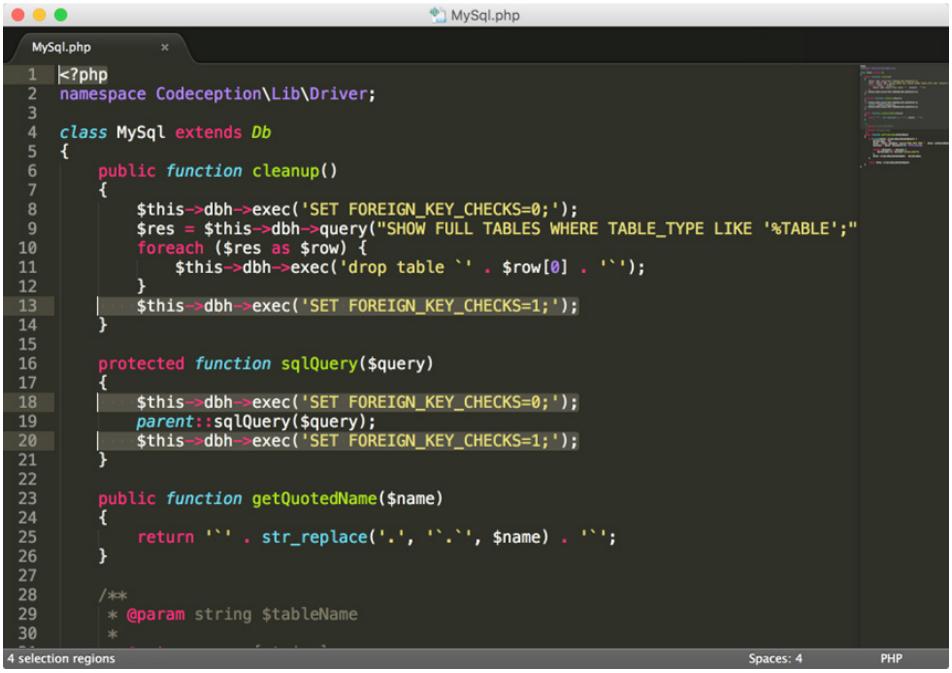


Bild 3.14 Mehrfachauswahl in Sublime Text

3.2.2 Entwicklungsumgebung

3.2.2.1 Eclipse

Eclipse gehört zu den wohl bekanntesten *IDEs* und hat seine Wurzeln in der Entwicklung von auf Java basierenden Programmen. Durch seine Erweiterbarkeit und der damit einhergehenden Unterstützung für weitere Programmiersprachen ist *Eclipse* aber auch als Umgebung zur Entwicklung mit anderen Sprachen geeignet. Dank der guten PHP-Unterstützung durch die *PHP Developer Tools* (kurz PDT) ist *Eclipse* inzwischen auch im PHP-Bereich ausgereift und gut nutzbar.

Eclipse basiert auf Java und ist dadurch unter allen genannten Betriebssystemen verfügbar. Es bietet umfangreiches Syntax-Highlighting, Unterstützung für Versionskontrollsysteme (*Git*, *SVN*, *CVS*, *Mercurial* etc.), Modellierungstools (*UML*), Code-Vervollständigung, Debugging (nach der Installation von *XDebug*) sowie unzählige weitere Funktionen. Leider merkt man *Eclipse* seinen großen Umfang auch in puncto Geschwindigkeit an, denn Eclipse benötigt gewaltige Mengen Arbeitsspeicher und ist ohne ein überaus performantes System nur mit viel Geduld nutzbar – im Bereich der kostenlosen *IDEs* ist *Eclipse* dennoch ohne Frage die erste Wahl.

Für die Nutzung in der auf PHP basierenden Anwendungsentwicklung empfiehlt sich die Installation mittels *All-In-One-Paket*, das Sie auf der Projektseite der *PHP Developer Tools* erhalten (*https://eclipse.org/pdt/#download*). Dieses Paket enthält sowohl die Eclipse-Grundver-

sion als auch die benötigten *PDT*, wodurch Sie sich die manuelle Nachinstallation sparen können.

HINWEIS: Der Abschnitt *Install Eclipse* der Anleitung *Setting up your workstation* im Joomla!-Dokumentationswiki enthält eine ausführliche Anleitung zur Konfiguration von Eclipse inklusive einer kurzen Bedienungseinführung: *https://docs.joomla.org/Configuring_Eclipse_for_joomla_development*

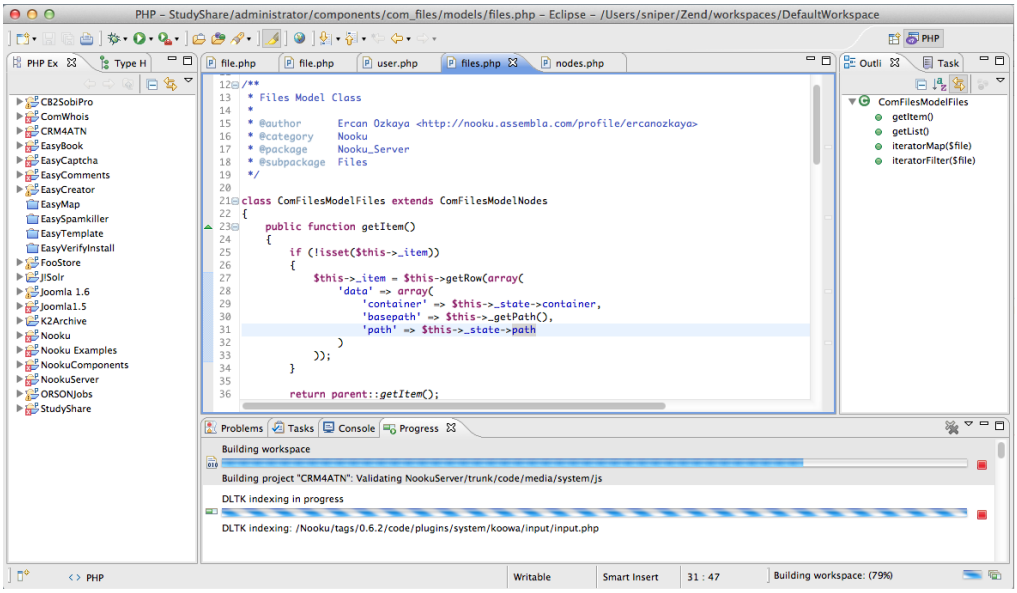


Bild 3.15 Eclipse

3.2.2.2 PhpStorm

Nachdem mich die oftmals nervenaufreibende Langsamkeit von *Eclipse* nahezu in den Wahnsinn getrieben hatte, habe ich mich auf die Suche nach einer alternativen IDE mit PHP-Unterstützung gemacht und bin auf *PhpStorm* (*www.jetbrains.com*) gestoßen. Diese Entwicklungsumgebung ist speziell für die Entwicklung mit PHP konzipiert, kann mit Features aufwarten, die *Eclipse* weit übertrumpfen (HTML5-Unterstützung, JavaScript-Debugging, PHPUnit-Unterstützung), und ist dabei schnell und genügsam. *PhpStorm* ist für alle genannten Betriebssysteme verfügbar, wird jedoch als kommerzielles Programm vermarktet, sodass man hier in eine entsprechende Lizenz investieren muss.

Diese Investition ist jedoch in jedem Fall lohnenswert, da PhpStorm die derzeit beste IDE im PHP-Bereich ist und sich zum Quasi-Standard in der Szene gemausert hat. Ein besonderes Highlight für Joomla-Entwickler ist dabei die integrierte Unterstützung für die Joomla!-API, die in dieser Form einzigartig im IDE-Bereich ist.

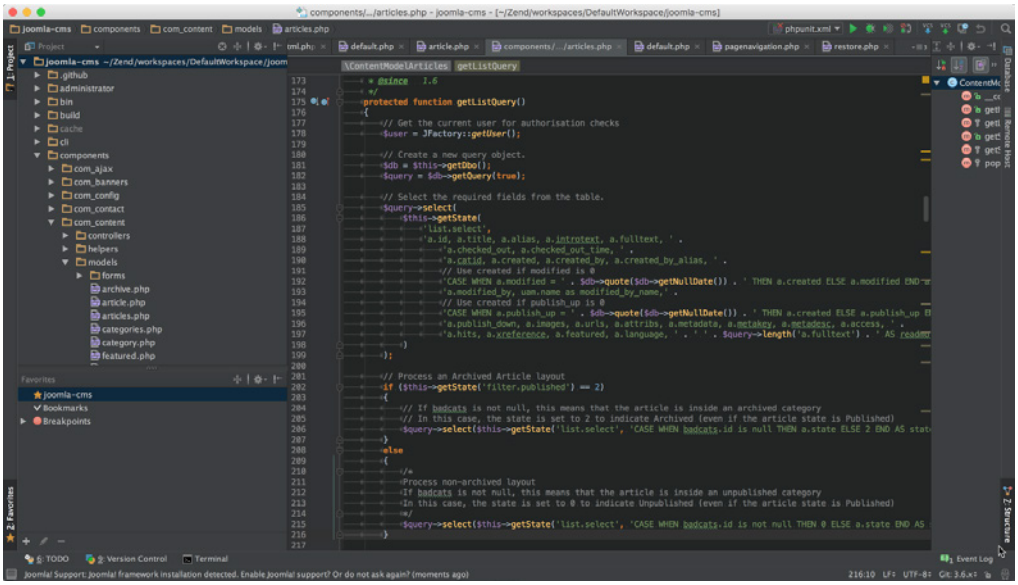


Bild 3.16 PhpStorm

PRAXISTIPP: Sie möchten eine Open-Source-Erweiterung für Joomla! programmieren? In diesem Fall können Sie auf einen besonderen Service der Firma *JetBrains* zurückgreifen und eine für Open-Source-Entwickler kostenlose Lizenz von *PhpStorm* erhalten.

3.3 Wahl des Browsers

Die Wahl des *Browsers* ist für Webentwickler und -designer ein essenzieller Schritt. Schließlich geht es hier um das wichtigste Arbeitsgerät, mit dem wir im weiteren Verlauf unter Umständen viele Stunden zubringen werden.

Letztendlich ist es wohl Geschmackssache, für welchen Browser man sich entscheidet, jedoch gibt es ein Kriterium, das bei der Wahl des Browsers beachtet werden sollte: Bei der Entwicklung empfiehlt es sich, einen Browser zu verwenden, der möglichst standardkonform bei der Darstellung von HTML und CSS ist. Denn durch die Optimierung des Codes in einem solchen Browser wird die Darstellung mit hoher Wahrscheinlichkeit auch bei anderen standardkonformen Browsern unseren Vorstellungen entsprechen – Sonderlösungen sind dann vermutlich nur noch bei älteren Browserversionen erforderlich. Welche Browser kommen hier also infrage? Man kann wohl guten Gewissens behaupten, dass sich der *Internet Explorer* in der Vergangenheit im Bereich der Standardkonformität nicht unbedingt mit Ruhm bekleckert hat, die neueren Versionen und insbesondere der Edge sind inzwischen jedoch sehr solide Tools geworden und dadurch durchaus einen genaueren Blick wert. Gängig ist

aber eher die Wahl eines aktuellen Webkit- (z. B. *Google Chrome*) oder Gecko-basierenden (z. B. *Mozilla Firefox*) Browsers, da diese durch ausgereifte Entwicklerwerkzeuge und gute Erweiterbarkeit viel Arbeit abnehmen. Mit Vorsicht zu genießen ist inzwischen leider der Safari-Browser, da dieser oftmals mit sehr alten Versionen der Webkit-Engine arbeitet, was der Standardkonformität nicht unbedingt zuträglich ist.

Zu enorm nützlichen Werkzeugen werden die meisten Browser aber erst nach der Installation einiger Erweiterungen, die speziell auf die Bedürfnisse von Webdesignern und -entwicklern abgestimmt sind. Meine persönlichen Favoriten möchte ich dabei kurz, aufgeteilt nach Browser, in der folgenden Tabelle vorstellen.

Erweiterung	Beschreibung
Google Chrome	
ChromePHP	Ermöglicht es, durch die Nutzung der gleichnamigen PHP-Bibliothek, direkt aus dem PHP-Code heraus Nachrichten an die Konsole der Chrome-Entwicklertools zu schicken. Ist daher ein äußert nützliches Debugging-Tool.
Awesome Screenshot	Erlaubt es, direkt im Browser Screenshots zu erstellen. Sinnvoll, um z. B. einen Darstellungsfehler zu dokumentieren.
Mozilla Firefox	
Firebug	Rüstet im Firefox die Funktionen der Entwicklertools nach, die in Webkit-basierenden Browsern (Chrome, Safari) standardmäßig vorhanden sind. Unverzichtbares Tool im Firefox.
FirePHP	**Siehe ChromePHP**
Web Developer Tools	Verankern sich mit Toolbar und Kontextmenü im Browser und erlauben uns, häufig ausgeführte Aufgaben wie das Deaktivieren von JavaScript, das Validieren der Seite, die Suche nach fehlenden Bildern oder das Verkleinern des Browserfensters mit nur einem Mausklick zu starten.
PageSpeed	Erlaubt die Analyse der Ladegeschwindigkeit der Seite und gibt zahlreiche Tipps zur Optimierung

3.3.1 Nutzung der Chrome-Entwicklertools

Wie werden die angesprochenen Entwicklertools für Firefox (nachzurüsten durch Firebug) bzw. Chrome und Safari (integriert) denn nun in der Praxis eingesetzt? Das möchte ich Ihnen kurz anhand von *Google Chrome* und einer kleinen Beispielseite zeigen. Die Entwicklertools werden über den Menüpunkt *Anzeigen* > *Entwickler* > *Entwickler-Tools* geladen und positionieren sich normalerweise im unteren Teil des Browserfensters.

PRAXISTIPP: Da man die Entwicklertools ständig nutzt und das Klicken durch das Menü leider relativ langwierig ist, lohnt es sich, den jeweiligen Tastatur-Shortcut zu lernen, der sich je nach Browser und Betriebssystem unterscheidet.

Hier werden nun verschiedene Reiter angeboten, welche die verschiedenen Funktionen der Entwicklertools repräsentieren:

- **Elements:** zeigt den HTML-Code der Seite in einer übersichtlichen Baumstruktur sowie die darauf angewendeten CSS-Befehle, die Abmessungen und JavaScript-Listener.
- **Console:** JavaScript-Fehler- und -Ausführungskonsole. Erlaubt die manuelle Ausführung von JavaScript.
- **Sources:** listet alle geladenen Dateien wie Stylesheets, Bilder und JavaScripts auf und erlaubt weiterhin die Anpassung und das Debugging von Skripten.
- **Network:** zeigt Dateigrößen, Ladezeiten, Latenzen und Pfade der eingebundenen Dateien.
- **Timeline:** zeigt den Zeit- und Speicherbedarf der verschiedenen Rendering- und Scripting-Schritte.
- **Profiles:** gibt an, welcher Programmteil des Browsers welchen Speicher- und CPU-Bedarf hat.
- **Security:** prüft die Sicherheit der Seite, wie z. B. das verwendete SSL-Zertifikat.
- **Resources:** zeigt die geladenen Dateien (siehe Sources) sowie die von der Seite genutzten Ressourcen im Bereich Local Storage, IndexedDB, Cookies, Cache und Server Workers.
- **Audit:** erlaubt die Durchführung eines Ladezeiten-Checks und gibt anschließend einen ausführlichen Optimierungsbericht zurück.

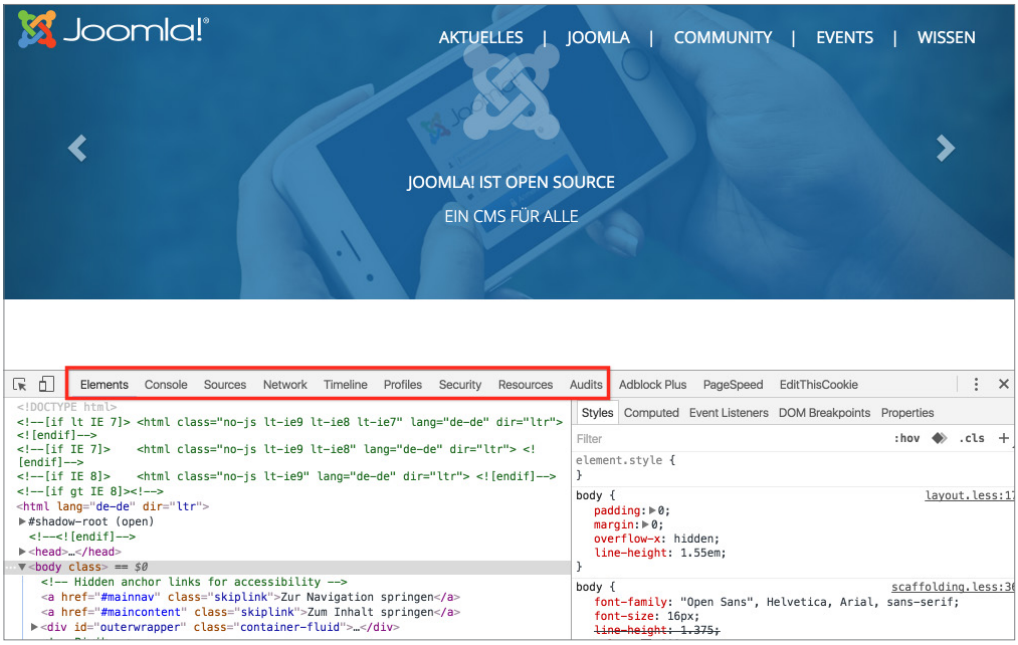


Bild 3.17 Chrome-Entwicklertools mit den verschiedenen, hier hervorgehobenen Funktionen

Für uns als Entwickler ist der Bereich *Profiles* relativ uninteressant, alle anderen Funktionen enthalten jedoch nützliche Werkzeuge zur Gestaltung unserer Seite.

Zuerst wollen wir uns hier den Bereich *Elements* anschauen, der das manuelle Browsen durch den HTML-Code erlaubt und dabei die zugeordneten CSS-Styles und Maße anzeigt. Gerade bei umfangreichen Seiten kann es jedoch lange dauern, bis man auf diese Weise das gewünschte Element gefunden hat. Deshalb gibt es die äußerst nützliche Funktion zum Auswählen eines bestimmten, im Fenster sichtbaren Objekts, die über das Cursor-Symbol in der oberen linken Ecke der Entwicklertools aufgerufen wird.

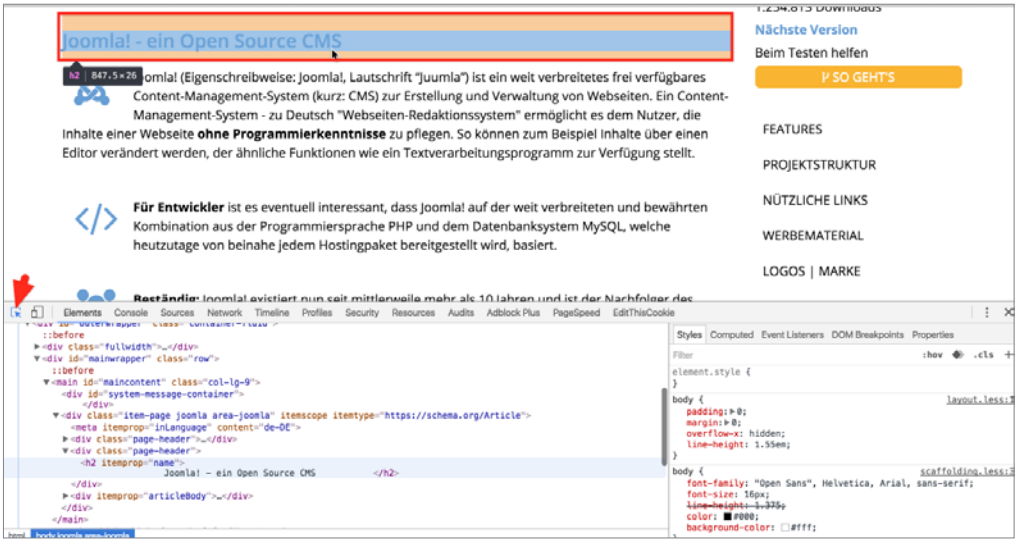


Bild 3.18 Auswahl eines Seitenelements mit der *Select an Element*-Funktion

Fährt man nun mit dem Mauszeiger über ein Element der Seite, wie zum Beispiel in Bild 3.18 die Seitenüberschrift, zeigt uns der Browser den HTML-Tag (sowie, falls vorhanden, etwaige CSS-Klassen und -IDs) und die Maße des jeweiligen Elements, die zudem grafisch durch eine farbige Box hervorgehoben werden. Klickt man nun auf das entsprechende Element, so wird es in den Entwicklertools ausgewählt, wodurch zum Ersten der entsprechende HTML-Tag im Baum markiert wird und zum Zweiten die entsprechenden, auf das Element angewendeten CSS-Befehle in der rechten Spalte der *Entwicklertools* erscheinen.

Der HTML-Code kann dabei nicht nur betrachtet, sondern durch einen Doppelklick auf das jeweilige Element auch temporär verändert werden, um beispielsweise einen anderen Überschrifttyp auszutesten, der ansonsten erst aufwendig per FTP verändert werden müsste. Die entsprechenden Änderungen sind dabei im Browser sofort sichtbar, werden aber beim Reload der Seite wieder zurückgesetzt.

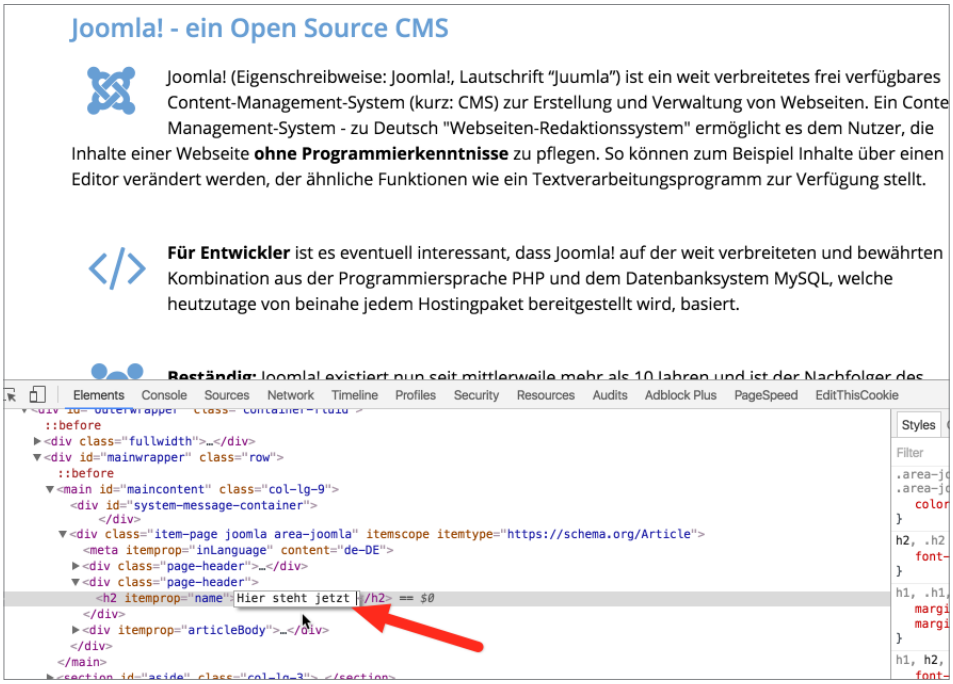


Bild 3.19 Editierung des HTML-Codes mit den *Entwicklertools*

Ähnliche Funktionen bieten die *Entwicklertools* nun auch für die Editierung des CSS-Codes in der rechten Spalte, in der alle angewendeten CSS-Befehle des jeweiligen Elements angezeigt werden. Die entsprechenden Befehle sind dabei nach der jeweiligen Fundstelle im CSS-Code gruppiert, zu der immer der zugehörige Dateiname inklusive Zeilennummer angegeben ist. Die Attribute lassen sich, analog zum Bearbeiten des HTML-Codes, mittels Doppelklick temporär editieren oder über die Checkbox am rechten Rand der Zeile ganz deaktivieren, wobei auch hier jede Änderung sofort im Browser angezeigt wird.

Bild 3.20 Auflistung und Änderung des CSS-Markups

Insbesondere diese Funktion zur Änderung der CSS ist ein für mich unverzichtbares Tool zur Arbeitserleichterung, da ich auf einen Blick sehen kann, welche CSS-Eigenschaften an welcher Stelle des Codes angewendet oder vererbt werden, wodurch sehr viel aufwendige Sucharbeit erspart bleibt.

3.4 FTP-Client

Zum Transfer der fertigen Seite auf einen angemieteten Webspace wird ein sogenannter FTP-Client benötigt. Für die drei bekanntesten Betriebssysteme Windows, Linux und OS X ist dafür der freie FTP-Client *Filezilla* verfügbar, der diese Aufgabe mit Bravour meistert.

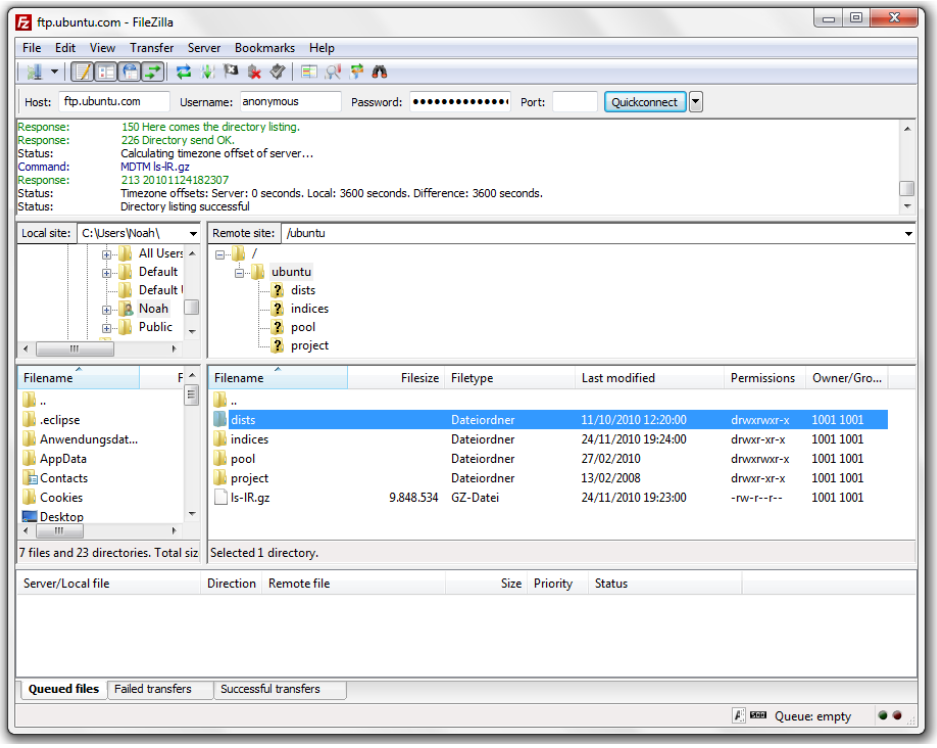


Bild 3.21 Filezilla

PRAXISTIPP: Unter OSX unterstützt *Filezilla*, aufgrund seiner Cross-Plattform-Kompatibilität, leider viele Funktionen des Betriebssystems wie Drag & Drop auf das Dock-Icon nicht, weshalb ich – insbesondere bei regelmäßiger Nutzung – zur Installation eines nativen FTP-Clients wie *Transmit* (*http://www.panic.com/transmit*) raten möchte.

3.5 Passwort-Manager

Als Webmaster, Entwickler und Designer kommen wir tagtäglich mit einer Vielzahl an Zugangsdaten und Passwörtern in Berührung: FTP, MySQL, Joomla!, der Kundenbereich des Webhosters und diverse Online-Dienste seien hier beispielhaft aufgezählt.

Um dieser Flut Herr zu werden, gehen viele Nutzer dazu über, sich ein einheitliches „Standardpasswort“ auszudenken und dieses dann mehrfach zu verwenden. Im Bereich der IT-Sicherheit kann dieser Fall getrost als Worst-Case-Szenario bezeichnet werden: Wird einer der Dienste, bei denen sich der Benutzer angemeldet hat, erfolgreich gehackt, so hat der Angreifer Zugriff auf das Klartext-Passwort und somit Zugang zu allen weiteren Diensten.

Eine gern genutzte Alternative stellen Excel- oder Wordlisten auf dem eigenen Rechner dar, in der alle – dann hoffentlich zufallsgenerierten – Passwörter, feinsäuberlich abgelegt sind. Der Nachteil dieses Ansatzes ist jedoch, dass die Passwortliste unverschlüsselt ist und ein Angreifer, z. B. über einen Trojaner, in den Besitz der kompletten Liste kommen kann.

Ich möchte daher für jeden Internetnutzer allgemein, für Webmaster, –designer und –entwickler aber ganz besonders, die Nutzung eines sog. Passwort-Managers empfehlen. Die Funktionsweise ist dabei stets dieselbe: Nach der Installation des Passwort-Managers vergibt man ein Master-Passwort, das den Zugriff auf die Passwörter regelt, die im Manager gespeichert sind. Dieses Master-Passwort wird dabei gleichzeitig dafür genutzt, die dort gespeicherten Passwörter mit einer sicheren Verschlüsselungstechnologie zu verschlüsseln, sodass ein Angreifer selbst mit Zugriff auf die Tresor-Datei keinen Zugriff auf die Klartext-Passwörter erlangen kann.

Gängige Tools sind dabei die OpenSource-Lösung Keypass sowie die beiden kommerziellen Anwendungen 1Password und Lastpass. Ich kann Ihnen an dieser Stelle nur sehr nachdrücklich dazu raten, ein solches Tool einzusetzen.

4 Installation

Jetzt sind endlich alle Vorbereitungen abgeschlossen und wir können mit der Installation unserer neuen Joomla!-Seite starten. Dabei ist zu unterscheiden, ob wir die Installation direkt auf einem Webserver oder in der lokalen Umgebung (XAMPP, MAMP, LAMP) durchführen wollen.

4.1 Installation in der lokalen Umgebung

Wir starten die Installation unserer Joomla!-Umgebung mit dem Download der aktuellsten Joomla!-Version 2.5 auf den Desktop. Dabei sollten Sie stets auf die offizielle Version zurückgreifen, die Sie auf *joomla.org* zum Download finden, und keinesfalls externe Download-Portale oder den Softwaremanager Ihres Webhosters nutzen, da es sich bei diesen Quellen oftmals um veraltete oder modifizierte Joomla!-Versionen handelt, die im weiteren Betrieb zu Problemen führen. Das entsprechende Archiv entpacken wir nun auf dem Desktop und benennen den entsprechenden Ordner um, damit er einen aussagekräftigeren Namen erhält – im Zweifelsfall bietet sich hier der entsprechende Titel des Projekts an, für das wir die jeweilige Seite erstellen wollen. Ich wähle „JoomlaBuch“ als Ordnername und verschiebe den fertigen Ordner in den sog. „Docroot“ meines lokalen Webservers (Windows: *C:\xampp\htdocs*, OS X: *Programme/MAMP/htdocs*, Linux: */var/www*).

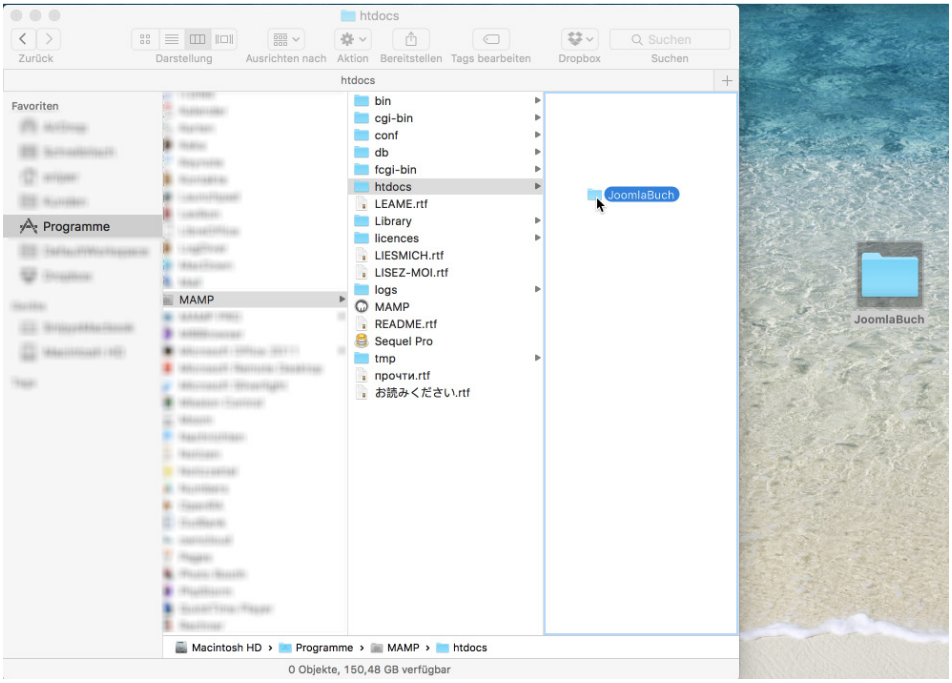


Bild 4.1 Verschieben des Ordners *JoomlaBuch* in das *htdocs*-Verzeichnis des Webservers unter OS X

Nun legen wir noch eine leere MySQL-Datenbank für unsere Installation an, indem wir im Browser das Tool phpMyAdmin aufrufen (XAMPP: *http://localhost:8888/xampp/phpMyAdmin*, MAMP: *http://localhost:8888/phpMyAdmin*, Linux: *http://localhost:8888/phpMyAdmin*) und darin, nach dem Login als Root, den Reiter *Datenbanken* öffnen. Hier legen wir im unteren Bereich der Seite den Namen für die neue Datenbank an und speichern diese durch den Klick auf ANLEGEN.

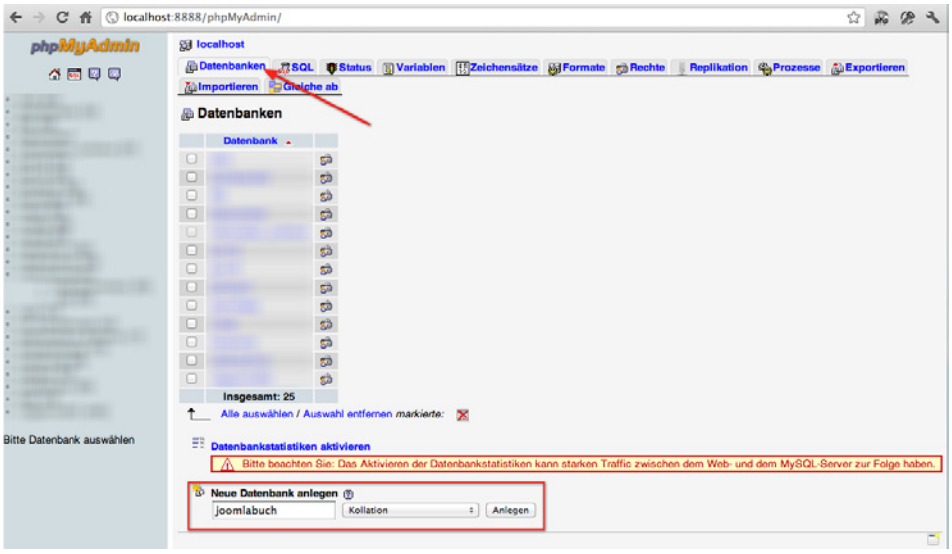


Bild 4.2 Erstellung einer lokalen MySQL-Datenbank mittels phpMyAdmin

Nun können wir die Joomla!-Installation durch den Aufruf von *http://localhost:8888/JoomlaBuch* im Webbrowser starten – sollten Sie beim Umbenennen des Ordners einen anderen Projektnamen gewählt haben, so müssen Sie diesen im angegebenen Link entsprechend anpassen.

Es öffnet sich der Joomla!-Installationsprozess, der mit der Auswahl der Sprache beginnt, die während der weiteren Installation genutzt werden soll. Direkt unterhalb der Sprachauswahl sehen Sie nun im Optimalfall ein Formular, mit dem die ersten Grunddaten der Joomla!-Installation erfasst werden – es sei denn, Ihre lokale Umgebung (bzw. Ihr Provider bei der Installation auf einem Webspace) erfüllt die nötigen Mindestvoraussetzungen für die Installation von Joomla! nicht. In diesem Fall erhalten Sie, wie im Bild 4.3 sichtbar, eine Auflistung der zu erfüllenden Parameter mit einer entsprechenden farblichen Hervorhebung.

Bild 4.3 Prüfung der Serverumgebung während der Installation

Joomla! prüft dabei diverse Voraussetzungen, die in der folgenden Tabelle 4.1 dokumentiert sind.

Tabelle 4.1 Benötigte Einstellungen in der Webserverumgebung

Parameter	Empfohlener Wert	Beschreibung
Benötigte Einstellungen		
PHP-Version >= 5.3.10	Ja	Prüft, ob die von Joomla! benötigte PHP-Version 5.3.10 (oder neuer) installiert ist
Magic Quotes GPC aus	Ja	Prüft, ob die PHP-Option magic_quotes_gpc ausgeschaltet ist
Register Globals aus	Ja	Prüft, ob die PHP-Option register_globals ausgeschaltet ist
Zlib-Kompressionsunterstützung	Ja	Prüft, ob PHP mit der Option `--with-zlib` kompiliert wurde
XML-Unterstützung	Ja	Prüft, ob PHP ohne die Option `--disable-xml` kompiliert wurde
Datenbank-Unterstützung	Ja	Prüft, ob PHP mit mindestens einem von Joomla! unterstützten Datenbanktreiber (z. B. mysqli) kompiliert wurde
MB-Sprache ist Standard	Ja	Prüft, ob die PHP-Direktive *mbstring.language* dem Standardwert „neutral“ entspricht
MB String overload ist deaktiviert	Ja	Prüft, ob die PHP-Direktive *mbstring.func_overload* deaktiviert ist
INI-Parser-Unterstützung	Ja	Prüft, ob die PHP-Funktionen `parse_ini_string()` und `parse_ini_file()` verfügbar sind. Falls nicht, liegt das im Regelfall daran, dass sie vom Hoster in der php.ini über die *disable_functions*-Direktive deaktiviert wurden.
JSON-Support	Ja	Prüft, ob die PHP-Funktionen `json_encode()` und `json_decode()` verfügbar sind oder in der php.ini durch die *disable_functions*-Direktive deaktiviert wurden
configuration.php nicht schreibgeschützt	Ja	Prüft, ob es möglich ist, die Datei *configuration.php* im Hauptverzeichnis von Joomla! zu erstellen. Sollte dies nicht der Fall sein, so muss die Rechtestruktur des Servers korrigiert werden (siehe Abschnitt 19.1.1, „Das www-run-Problem“).
Empfohlene Einstellungen		
Safe-Mode	Aus	Prüft den Status der Konfigurationsvariablen *safe_mode* in der php.ini
Fehler anzeigen	Aus	Prüft, ob Fehler, die bei der Ausführung des Skripts auftreten, auf dem Bildschirm ausgegeben werden. Zuständig dafür ist die PHP-Direktive *display_errors* in der php.ini.
Dateien hochladen	An	Überprüft, ob Dateien auf den Server hochgeladen werden können. PHP-Direktive: *file_uploads*.
Magic-Quotes-Laufzeit	Aus	Prüft, ob die PHP-Direktive *magic_quotes_runtime* aktiviert ist, die bei schlecht programmierten Erweiterungen zu unerwarteten Fehlern führen kann
Gepufferte Ausgabe	Aus	Prüft, ob die PHP-Direktive *output_buffer* aktiv ist

Parameter	Empfohlener Wert	Beschreibung
Automatischer Sitzungsstart	Aus	Prüft, ob die PHP-Direktive *session.auto_start* aktiv ist
Standard-ZIP-Unterstützung	An	Prüft, ob PHP mit der Option `--with-zlib` kompiliert wurde

Sollte einer der Parameter unter *Installationsprüfung* dabei nicht den Vorgaben entsprechen, so ist es leider nicht möglich, Joomla! auf diesem System zu betreiben, weshalb der Installationsprozess an dieser Stelle stoppt und Joomla! die erneute Prüfung der Umgebung anbietet.

Die *empfohlenen Einstellungen* sollten nach Möglichkeit den Vorgaben entsprechen, beeinflussen allerdings nicht den Betrieb des Joomla!-Kerns, weshalb die Installation problemlos möglich ist.

Sollte einer der genannten Parameter in unserer lokalen Umgebung nicht zu den Vorgaben passen, so können wir diesen durch eine entsprechende Änderung in der *php.ini* anpassen – sollte dieses Problem jedoch auf dem Webspace des Hosters auftreten, so können die nötigen Änderungen im Regelfall nur durch den Hoster selbst vorgenommen werden, da die *php.ini* außerhalb des für uns beschreibbaren Bereichs des Servers liegt.

PRAXISTIPP: Es gibt einen kleinen Trick, die entsprechende *php.ini*-Datei zu finden, die für die PHP-Instanz verantwortlich ist, in der unsere Joomla!-Installation läuft. Dafür erstellen wir als Erstes eine Datei namens *info.php* mit dem Inhalt `<?php phpinfo(); ?>` und legen diese ins Hauptverzeichnis unserer Joomla!-Installation. Anschließend rufen wir die Datei über den Browser auf und können aus der `phpinfo()`-Ausgabe die Pfadangabe zu *php.ini* ablesen.

Bild 4.4 Ausgabe der `phpinfo()` mit Pfadangabe zur ***php.ini***

Sind alle Voraussetzungen erfüllt, können wir mit der Installation fortfahren. Bevor wir dabei die von Joomla! benötigten Angaben eintragen, werfen wir noch einen Blick in die GNU General Public License, kurz GNU GPL, auf die Joomla! im oberen Bereich des Bildschirms hinweist. Bei der GPL handelt es sich um eine sog. Freie Software-Lizenz mit Copyleft, die jedem Nutzer erlaubt, den Quellcode eines Werks, das unter der GPL steht,

- beliebig in unveränderter Form zu vervielfältigen und zu verbreiten,
- beliebig zu verändern und zu verbreiten, solange die modifizierte Version ebenfalls unter den Bedingungen der GPL verbreitet wird.

Es ist dabei ein weit verbreiteter Irrtum, dass GPL-lizenzierte Software per se kostenlos verfügbar sein muss, denn dies wird von der Lizenz nicht vorgegeben – im Lizenztext wird sogar explizit die Möglichkeit zur kommerziellen Vermarktung eingeräumt.

Wir behalten dieses Wissen nun erst einmal im Hinterkopf und fahren, nach der Lektüre der Lizenzbedingungen, mit der Installation fort.

Im nächsten Schritt, fordert uns Joomla! auf, diverse Grunddaten anzugeben, die für die Installation benötigt werden.

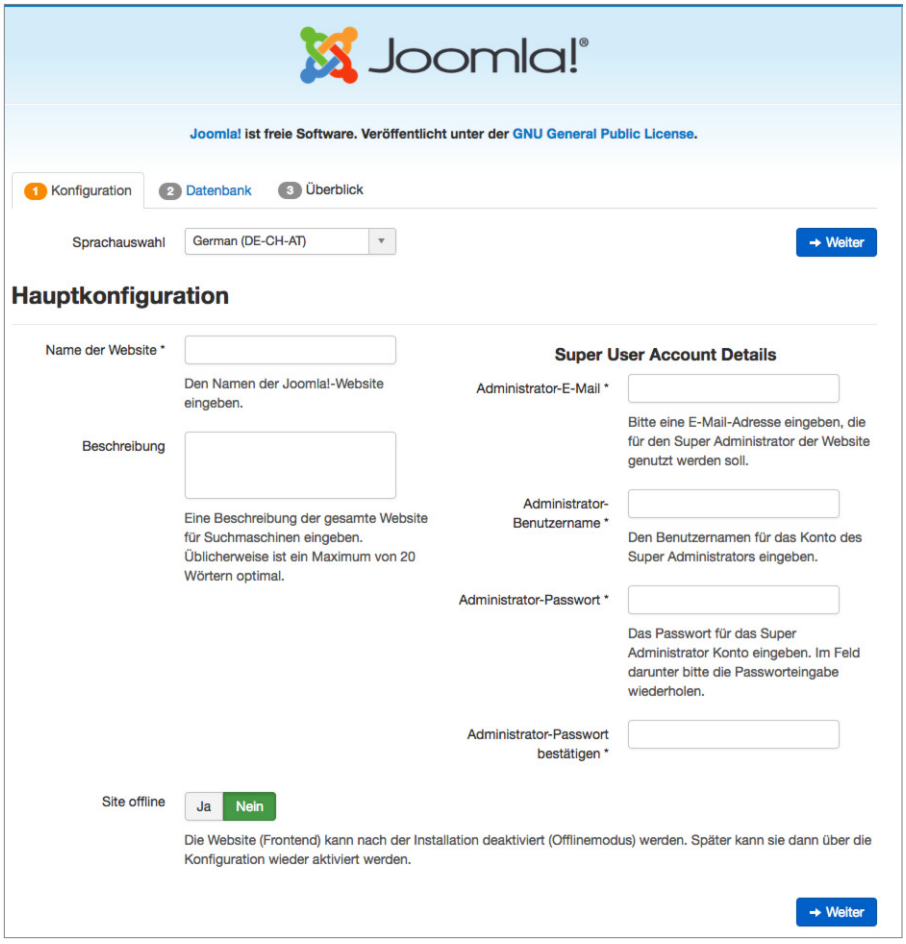


Bild 4.5 Erfassung der Grunddaten der Installation

Eine Beschreibung der einzelnen Parameter finden Sie in der folgenden Tabelle:

Tabelle 4.2 Konfigurationsparameter von Schritt 1 im Installationsprozess

Parameter	Erklärung
Name	Name der Website, der später als Seitentitel der Startseite, als Überschrift im Administrationsbereich und als Absendername von E-Mails verwendet wird. Kann später im Administrationsbereich verändert werden.
Beschreibung	Meta-Beschreibungstext der Seite für Suchmaschinen
Site Offline	Versetzt die Seite nach Abschluss der Installation in den *Offline-Modus*, damit sie nur vom Administrator betrachtet werden kann
Administrator-E-Mail	E-Mail-Adresse des Administrators der Seite
Administrator-Benutzername	Ihr gewünschter Benutzername
Administrator-Passwort	Administrator-Passwort (Empfehlung: mindestens zehn Zeichen; Klein- und Großbuchstaben, Zahlen sowie Sonderzeichen; keine Umlaute)

Wir tragen hier die entsprechenden Daten ein und setzen die Installation mit einem Klick auf *Weiter* fort.

Nun wird die Konfiguration der Datenbankverbindung vorgenommen, bei der die folgenden Parameter einzustellen sind.

Tabelle 4.3 Erklärung der Parameter der Datenbankverbindung

Parameter	Erklärung
Datenbanktyp	Derzeit unterstützt Joomla! sechs verschiedene Arten der Anbindung an den Datenbank-Server: *MySQLi*: verbesserte (**i**mproved) MySQL-Erweiterung für PHP – **empfohlen** *MySQL*: ältere MySQL-Erweiterung mit schlechterer Performance, wurde mit PHP7 entfernt *MySQL (PDO)*: MySQL-Erweiterung auf Basis der PDO API, etwas langsamer als MySQLi *PostgreSQL*: Unterstützung für den freien Datenbankserver PostgreSQL *Microsoft SQL-Server*: Unterstützung für den MS SQL Server *Microsoft SQL-Azure*: Unterstützung für MS SQL Server in Azure-Umgebungen Zu beachten ist hierbei, dass die Unterstützungen für PostgreSQL sowie den Microsoft SQL Server nur bei sehr wenigen Nutzern im Einsatz sind und daher als fehleranfällig und instabil gelten. Aktuell wird darüber diskutiert, die Unterstützung für MySQL in kommenden Versionen ganz zu entfernen.
Servername	Name (oder IP-Adresse) des Datenbankservers. Falls nicht anders angegeben, ist dies im Regelfall „localhost“.
Benutzername	Benutzername für die Datenbankverbindung. In lokalen Umgebungen häufig „root“, auf dem Webspace ist der Name dem Control-Panel des Hosters zu entnehmen.
Passwort	Passwort für die Datenbankverbindung. In der lokalen Umgebung normalerweise entweder „root“ oder gar nicht gesetzt und kann dann leer bleiben. Auf dem Webspace ist das Passwort im Control-Panel des Hosters zu vergeben.
Datenbankname	Name der Datenbank, die wir vor Installationsbeginn erstellt haben.

Parameter	Erklärung
Tabellenpräfix	Ermöglicht es, mehrere Joomla!-Installationen in nur einer Datenbank zu betreiben, da es als Zeichenkette vor den eigentlichen Tabellennamen gesetzt wird. Bis einschließlich Joomla! 1.5 standardmäßig auf „jos_“ gesetzt, seitdem zufallsgeneriert, weshalb im Normalfall keine Anpassung erforderlich ist.
Alte Datenbanktabellen löschen	Erlaubt es, bereits vorhandene Tabellen einer alten Joomla!-Installation (mit gleichem Tabellenpräfix) in der gewählten Datenbank entweder zu löschen oder mit dem neuen Präfix „bak_“ zu versehen und dadurch zu sichern

Von Zeit zu Zeit kommt es vor, dass Joomla! den nächsten Installationsschritt mit nicht sehr aussagekräftigen Fehlermeldungen wie „Verbindungsfehler“ verweigert. In einem solchen Fall ist der Fehler eigentlich immer bei falsch eingetragenen Verbindungsdaten (Buchstabendreher, Leerzeichen oder Umlaute im Passwort oder Benutzernamen, falscher Servername) zu suchen.

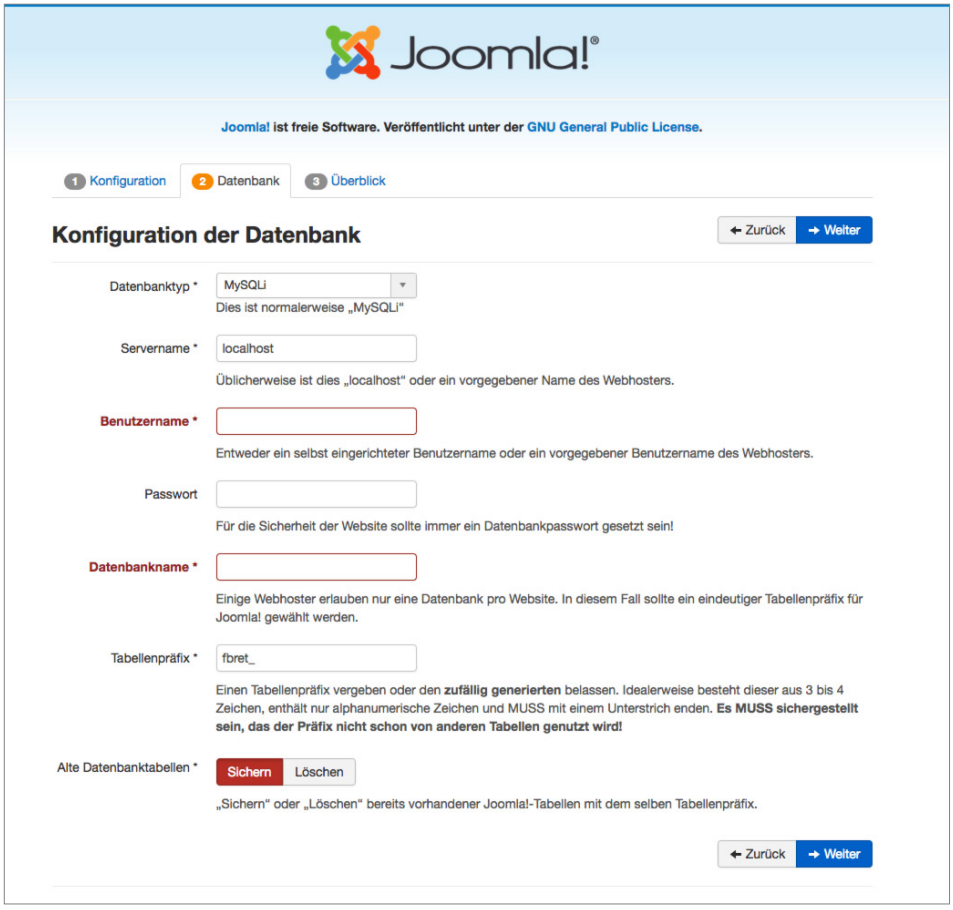


Bild 4.6 Konfiguration der Datenbankverbindung während der Installation

Im Normalfall tragen wir hier aber einfach unsere Datenbankverbindungsdaten ein und fahren durch einen Klick auf WEITER mit der Installation fort.

In Schritt 3 der Installation fragt das System ab, mit welchem Beispieldatensatz wir die Installation fortsetzen möchten. Zur Auswahl stehen hierbei:

- Keine
- Englische (GB) Beispieldaten: Bloginhalte
- Englische (GB) Beispieldaten: Prospektinhalte
- Englische (GB) Beispieldaten: Standardinhalte
- Englische (GB) Beispieldaten: Joomla! erlernen
- Englische (GB) Beispieldaten: Testinhalte

Im Regelfall sollten Sie hier die Option *Keine* wählen, da die in Joomla! integrierten Beispieldatensätze für produktive Websites ohnehin uninteressant sind und dann händisch wieder entfernt werden müssten. Ein weiterer Vorteil ist, dass die Installation ohne Beispieldaten es erlaubt, Joomla direkt für mehrsprachige Websites einzurichten.

Bild 4.7 Auswahl des zu installierenden Beispieldatensatzes

Nach der Auswahl des Beispieldatensatzes können wir nochmal einen letzten Blick auf die von uns eingegebenen Daten werfen und die Installation anschließend mit dem Klick auf *Installieren* starten.

Anschließend wird die Installation von Joomla! durchgeführt und nach Abschluss des Prozesses beglückwünscht uns das System zur Bewältigung dieser Aufgabe. Als letzter Schritt ist nun noch nötig, dass wir das Verzeichnis */installation* im Joomla!-Verzeichnis löschen, damit niemand sonst mehr das Installationstool nutzen und damit Schaden anrichten kann. Die Löschung können wir bequem über den entsprechenden, prominent platzierten Button (siehe Bild 4.8) durchführen.

Nun können wir unsere neue Seite über den Klick auf den Button *Website* aufrufen.

Bild 4.8 Erfolgreicher Abschluss der Installation

Bei der Installation von Joomla! gibt es jedoch zwei Sonderfälle zu berücksichtigen, auf die ich im Folgenden noch kurz eingehen möchte.

4.1.1 Sonderfall 1: der FTP-Modus

Der erste Sonderfall begegnet uns, wenn wir Joomla! in einer Umgebung installieren, in der das CMS aufgrund des sog. www-run-Problems (siehe Abschnitt 19.1.1, „Das www-run-Problem") keinen Schreibzugriff auf seine eigenen Dateien hat. Das ist insofern problematisch, als dass dadurch viele Grundfunktionen des Systems (Schreiben der Konfigurationsdatei, Erweiterungsinstallation, Bildupload) nicht funktionieren würden. Um dieses Problem zu umgehen, prüft Joomla! im Verlauf der Installation automatisch, ob die Dateien beschreibbar sind und fügt im Fall der Fälle einen neuen Schritt 3 an die Eingabe der Datenbankdaten an.

In diesem Schritt bittet Joomla! darum, dass man die Daten des FTP-Accounts angibt, mit dem Joomla! z. B. beim Webhoster hochgeladen wurde. Joomla! nutzt dann diese Daten, um eine Verbindung mit dem FTP-Server aufzubauen und entsprechende Schreibzugriffe über diesen Umweg durchzuführen.

Leider sind die FTP-Zugriffe enorm langsam und unzuverlässig, weshalb die Nutzung des FTP-Modus nur im äußersten Notfall in Erwägung gezogen werden sollte – besser ist, hier bei der Hosterauswahl auf eine vernünftige Joomla!-Unterstützung zu achten, sodass das Problem erst gar nicht auftaucht.

Wenn man jedoch dann doch einmal vor dem genannten Problem steht, trägt man die entsprechenden FTP-Serverdaten ein und aktiviert die FTP-Funktion durch die Betätigung des Buttons *Ja* im entsprechenden Dialog (siehe Bild 4.9).

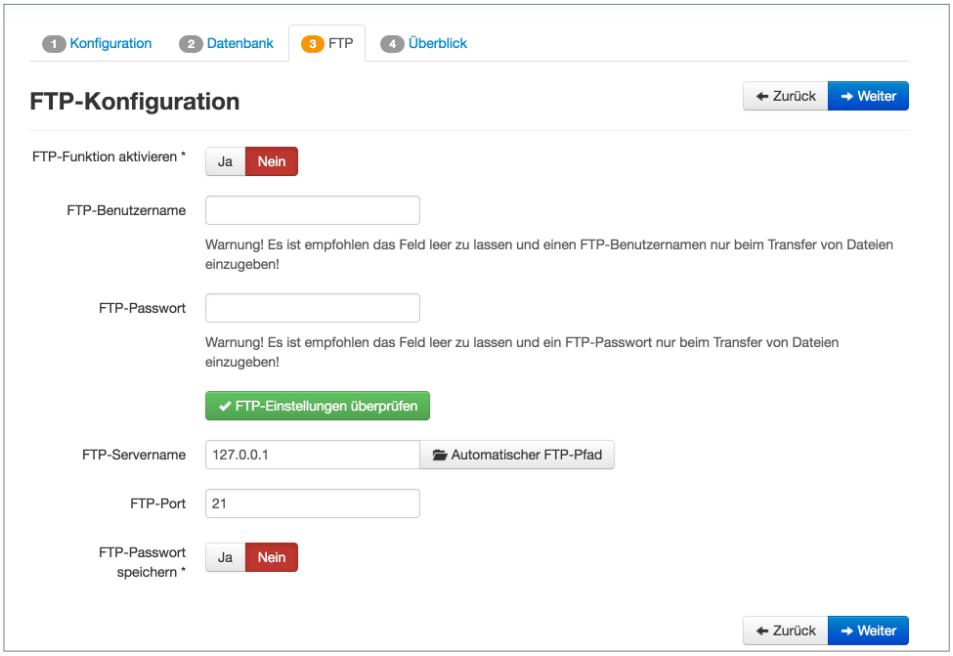
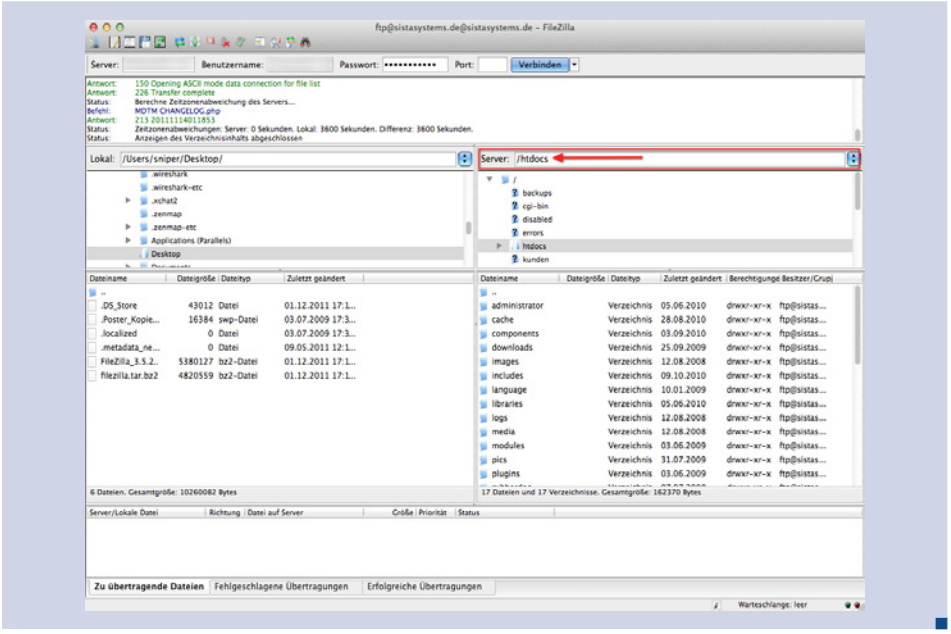


Bild 4.9 Eingabe der FTP-Zugangsdaten während der Joomla!-Installation

PRAXISTIPP: Der korrekte *FTP-Root-Pfad* lässt sich am einfachsten feststellen, indem man mit einem FTP-Client wie *Filezilla* zur entsprechenden Joomla!-Installation navigiert und dann den dort angegebenen Pfad auf den Server überträgt.



Die Angaben zu *FTP-Benutzername* und *FTP-Passwort* sollten in jedem Fall leer gelassen werden, da diese ansonsten im Klartext in der *configuration.php* gespeichert werden, was ein potenzielles Sicherheitsrisiko darstellen würde.

Wenn die Zugangsdaten für den FTP-Zugang korrekt konfiguriert sind, kann die Installation wieder mittels Klick auf WEITER fortgesetzt werden.

4.1.2 Sonderfall 2: mehrsprachige Installation

Wenn man bereits zum Zeitpunkt der Installation weiß, dass die Joomla!-Website mehrsprachig werden soll, sollte im letzten Schritt darauf geachtet werden, dass der Button *Verzeichnis „installation" löschen* (siehe Bild 4.8) nicht vorschnell betätigt wird. Der Installationsprozess kann hier nämlich über den Button *Extra Schritt: Sprachen installieren* direkt noch dafür genutzt werden, die benötigten Sprachdateien nachzuinstallieren und die neue CMS-Installation für die Verwendung von Mehrsprachigkeit vorzubereiten.

Dafür wählen Sie die gewünschten Zusatzsprachen aus (siehe Bild 4.10) und aktivieren im folgenden Schritt die Mehrsprachigkeitsfunktion (siehe Bild 4.11) und wählen die gewünschte Standardsprache aus.

Bild 4.10 Auswahl der zu installierenden zusätzlichen Sprachpakete

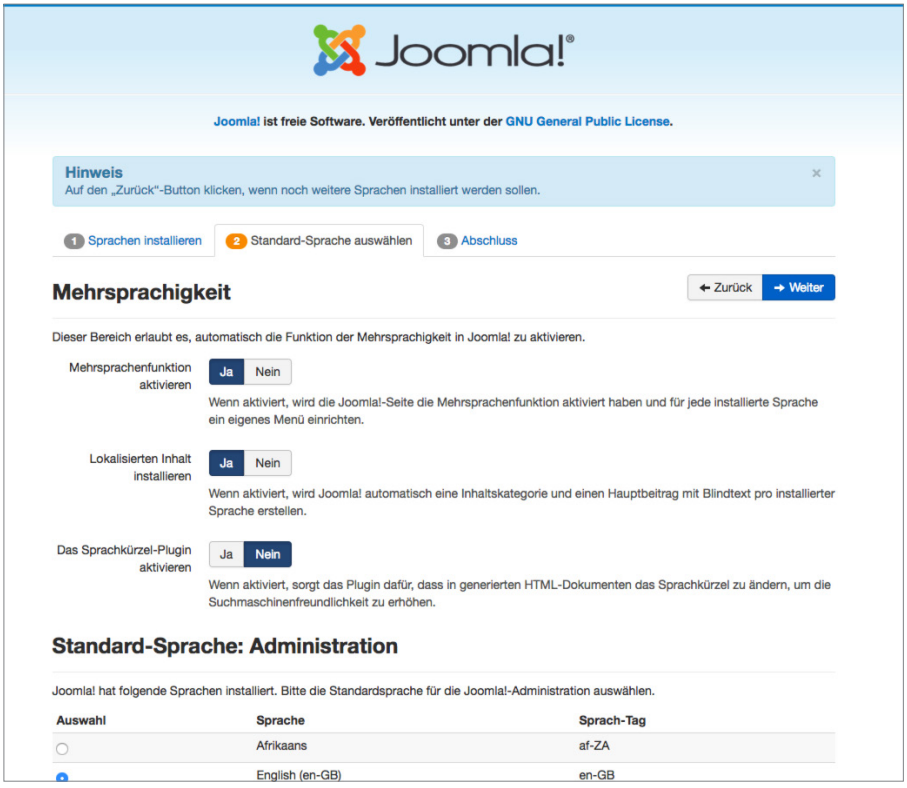


Bild 4.11 Konfiguration der Mehrsprachigkeitsfunktion während des Installationsprozesses

Anschließend löschen Sie auch hier das Verzeichnis */installation* durch Betätigung des entsprechenden Buttons.

Sie können diesen zusätzlichen Installationsschritt nicht nur dafür nutzen, die Mehrsprachigkeitsfunktion zu konfigurieren, sondern er bietet sich auch zur einfachen Installation des deutschen Sprachpakets bei einsprachigen Seiten an. Lassen sie hierfür einfach die Mehrsprachigkeitsfunktion (siehe Bild 4.11) deaktiviert und setzen Sie nur die deutsche Sprache als Standard.

4.2 Installation auf dem Webspace des Hosters

Die Installation auf dem Webspace des Hosters unterscheidet sich von der in Abschnitt 4.1 beschriebenen Installation in der lokalen Umgebung nur durch drei wesentliche Punkte:

Erstens ist es bei Webhostern in der Regel nicht möglich, die Serverkonfiguration so anzupassen, dass sie den Bedürfnissen von Joomla! entspricht. Deshalb sollten Sie schon vor Vertragsabschluss intensiv das Abschnitt 19.1, „Die Auswahl des richtigen Hosters“, lesen – dadurch ersparen Sie sich viel Ärger bei der späteren Nutzung.

Außerdem ist es notwendig, die Dateien mittels eines FTP-Clients wie *Filezilla* (siehe Abschnitt 3.4, „FTP-Client“) auf den Server des Hosters zu transferieren. Dafür öffnet man den FTP-Client, trägt die im Control-Panel (Confixx, Plesk, KIS) des Hosters hinterlegten FTP-Zugangsdaten ein und öffnet eine neue Verbindung.

Sie sollten – wann immer möglich – auf SFTP statt FTP zurückgreifen. FTP wickelt alle Datenübertragungen unverschlüsselt ab, sodass Angreifer z. B. Ihre FTP-Zugangsdaten abgreifen können. Bei SFTP, was von vielen großen Hostern und allen gängigen FTP-Clients unterstützt wird, erfolgt der Transfer verschlüsselt.

Nun transferiert man die Dateien der Joomla!-Installation, in der Regel mittels Drag & Drop, in den sog. *docroot*-Ordner des Webservers, der häufig als *htdocs* oder httpdocs benannt ist. Warum ins Docroot? Nun, ist eine Datei namens *blub.txt* direkt im Docroot *abgelegt*, so kann diese später über den Aufruf von *www.domain.tld/blub.txt* abgerufen werden; liegt die Datei hingegen im Unterordner *test* des Docroot, so wird sie später über die URL *www.domain.tld/test/blub.txt* abgerufen. Würden wir also Joomla! in einen Unterordner ablegen, so müssten wir später erst eine umständliche Weiterleitung einrichten, damit beim Aufruf von domain.tld auch tatsächlich unsere Installation erscheint.

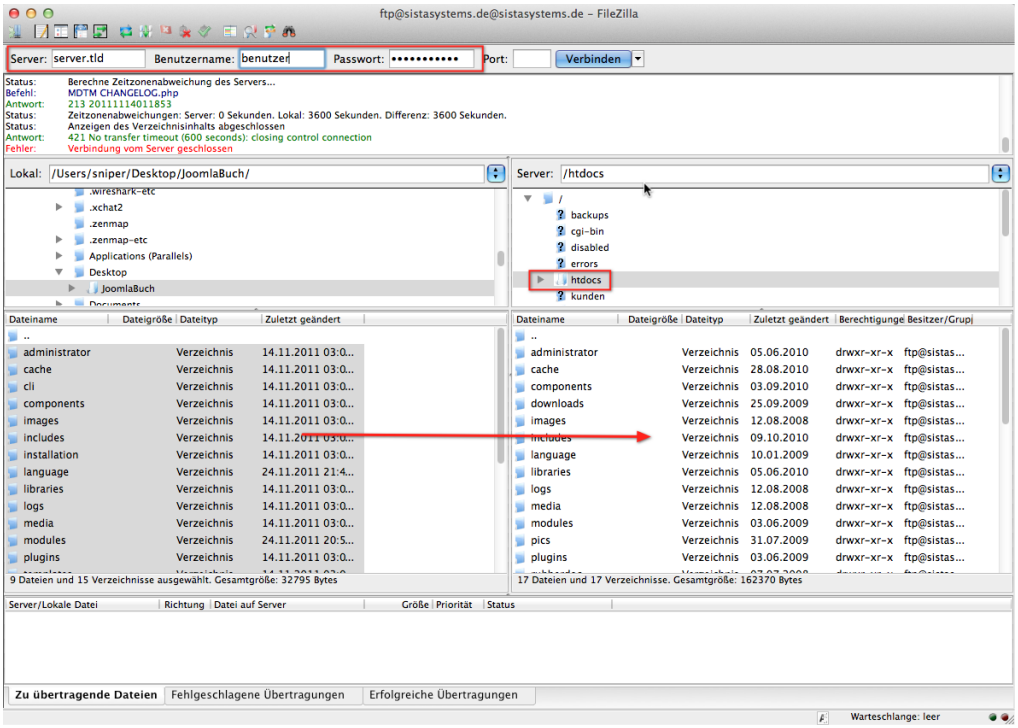


Bild 4.12 Transfer der Joomla!-Installation auf den Server mittels FileZilla

Nach dem Transfer ist es notwendig, im Control-Panel des Hosters eine neue MySQL-Datenbank sowie einen neuen, zugehörigen MySQL-Benutzer anzulegen. Anschließend kann die Installation durch den Aufruf von *www.domain.tld/installation* gestartet werden – die weiteren Installationsschritte sind dann Abschnitt 4.1 zu entnehmen.

4.3 Erste Handgriffe nach der Installation

Hiermit ist die Installation unserer Joomla!-Seite abgeschlossen. Standardmäßig sind jedoch einige Details des Joomla!-Pakets suboptimal und sollten deshalb unmittelbar nach der Installation angepasst werden.

4.3.1 Anpassung der robots.txt

Die standardmäßig bei Joomla! mitgelieferte *robots.txt*-Datei erlaubt es Suchmaschinen nicht, das Unterverzeichnis */cache* zu durchsuchen, wodurch CSS- und JavaScript- und Bilddateien, die in diesem Verzeichnis von einigen beliebten Erweiterungen abgelegt werden,

aus dem Suchmaschinenindex ausgeschlossen sind. Das hat z. B. den Nachteil, dass Suchmaschinen die Mobilfreundlichkeit der späteren Seite nicht mehr korrekt beurteilen können. Daher empfiehlt es sich, den entsprechenden Eintrag in der *robots.txt*, die sich im Hauptverzeichnis der Joomla!-Installation befindet, auszukommentieren, sodass die Datei nach der Editierung dem entsprechenden Ausschnitt in Listing 4.1 entspricht.

Listing 4.1 Ausschnitt der modifizierten *robots.txt*-Datei

```
User-agent: *
Disallow: /administrator/
Disallow: /bin/
#Disallow: /cache/
Disallow: /cli/
Disallow: /components/
Disallow: /includes/
Disallow: /installation/
Disallow: /language/
Disallow: /layouts/
Disallow: /libraries/
Disallow: /logs/
Disallow: /modules/
Disallow: /plugins/
Disallow: /tmp/
```

HINWEIS: Sollte Ihre Joomla!-Installation in einem Unterordner der Domain liegen, also nicht direkt über *www.domain.tld* erreichbar sein, so müssen Sie die *robots.txt* in das Wurzelverzeichnis der Domain schieben (*www.domain.tld/robots.txt*) und die in der Datei angegebenen Pfade entsprechend anpassen, da robot-Dateien gemäß dem entsprechenden Standard stets im *Docroot* der Domain zu finden sein müssen.

4.3.2 Leeren des Verzeichnisses /images

Nachdem der */images*-Ordner nun für Suchmaschinen zugänglich ist, empfiehlt es sich – insbesondere wenn die Seite später durch unerfahrene Nutzer administriert werden soll –, das Verzeichnis zu leeren und die dort abgelegten Beispieldaten zu entfernen. Andernfalls würden diese Daten an verschiedenen Stellen der Joomla!-Administration (Medien-Manager, Bild-Manager des Editors) immer wieder auftauchen und Verwirrung stiften. Lediglich die dort ebenfalls vorhandene *index.html* sollte aus Sicherheitsgründen erhalten bleiben.

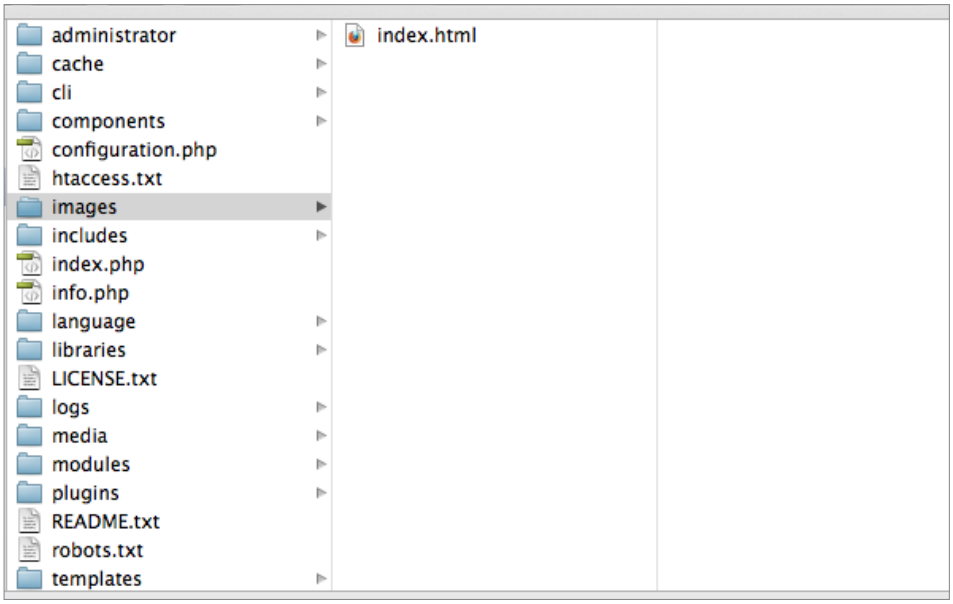


Bild 4.13 *images*-Verzeichnis nach dem Entfernen der Beispieldaten

HINWEIS: Dieser Schritt sollte ausgelassen werden, wenn während der Installation die Funktion *Beispieldaten installieren* genutzt wurde, da andernfalls die in den Beispieldaten hinterlegten Bilder nicht mehr angezeigt werden würden.

5 Grundlegende Begriffe und Architektur

Bevor wir nun mit dem Aufbau unserer Seite beginnen, möchte ich Ihnen zunächst einen kleinen Einblick in den Grundaufbau von Joomla! geben.

5.1 Grundlegende Begriffe

Wenn Sie als Neuling oder Quereinsteiger die Arbeit mit Joomla! in Angriff nehmen, werden Sie sich vermutlich mit einigen Begrifflichkeiten schwertun, da diese im Joomla!-*Administrationsbereich* nicht ausführlich erklärt sind. Daher möchte ich, bevor wir mit der Einrichtung der Seite starten, erst einmal einige wichtige Begriffe klären.

5.1.1 Backend/Frontend

Das CMS Joomla! teilt sich in seiner aktuellen Version in zwei wesentliche Bereiche, die sich *Frontend* und *Backend* nennen. Das *Frontend*, also die „Vorderseite" unseres Systems, ist dabei die eigentliche Website, die der Besucher der Seite mittels Browser aufruft.

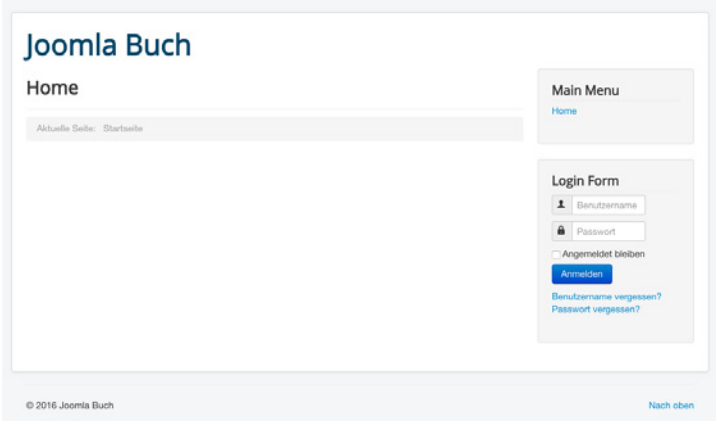


Bild 5.1 Frontend der in Abschnitt 4.1 installierten Joomla!-Seite