Realisierung eines Komplexpraktikums zur Erzeugung und Integration von Templates in ein CMS

Diplomarbeit

Eingereicht von

Toni Niemeier

16564

November 2011

Betreuer:

Prof. Dr. – Ing. Frank Zimmer **Hochschule Mittweida** Fachbereich EIT

Dipl. – Ing. (FH) Birger Jesch **Hochschule Mittweida** Fachbereich EIT

Inhaltsverzeichnis

Inhaltsverzeichnis 1
1 Einleitung 1
2 Vorbetrachtung 5
2.1 Joomla! 1.5 – Einführung 5
2.1.1 Begriffserklärung 5
2.1.2 Der Aufbau eines CMS- Systems 9
2.2 Aufgabenstellung 10
2.3 Anforderung an das Komplexpraktikum 11
3 Entwurf 12
3.1 Themenwahl für das Komplexpraktikum 12
3.2 Konzeptionierung des Templates 12
3.2.1 Auswahl des Layout zum Joomla Template 13
3.2.2 Das Screendesign zum Joomla-Template 14
3.2.3 Analyse vom Screendesign zum Webdesign/Template 17
3.2.4 Technische Vorrausetzungen für das CMS-Joomla! 19
3.2.5 Das Template-Gerüst 19
3.3 Konzeptionierung des Komplexpraktikums 22
3.3.1 Die Grundstruktur des Komplexpraktikums 22
4. Die Erzeugung des Komplexpraktikums 26
4.1 Erzeugung des Joomla!-Templates 26
4.1.1 Lokale Joomla! Installation 26
4.1.2 Joomla!-Backend Konfiguration 30
4.1.3 Die Dateien- und Verzeichnisstruktur für das Webprojekt 32
4.1.4 Das HTML-Gerüst 33
4.1.5 Das HTML-Gerüst 35

4.1.6 Die templateDetails.xml-Datei 40
4.1.5 Die Template-Installation 41
4.1.7 Erstellen der Webseitennavigation 42
4.1.8 Dynamische Inhalte erstellen 47
4.2 Erzeugung der Praktika 59
Abbildungsverzeichnis 61
Anhang A 62
Anhang B 63
Literarturverzeichnis 64

1 Einleitung

Content-Management-System - Hinter diesem Begriff verbirgt sich die Trennung von Content (Inhalten), Struktur und Design. Die Teilung in diese 3 verschiedenen Bausteine, kann während des Lebenszyklus‘ einer Webseite und sogar darüber hinaus entscheidende Vorteile mit sich bringen. Während heutzutage der Umgang mit Facebook, Myspace oder anderen Social-Media-Webseiten zum Alltag eines jeden gehört, wird die Wichtigkeit von CMS-Systemen immer bedeutender für den Anwender. Man kommt in der Regel, wenn man in einer Tätigkeit als Webentwickler steckt, kaum mehr daran vorbei, ein oder sogar mehrerer CMS-Systeme zu beherrschen. Die Anwender des World Wide Web sind es mittleiweile gewohnt überall und ganz einfach Inhalte in Form von Beiträgen, Bildern, Kommentaren, ja sogar Videos einzustellen und natürlich wollen sie diesen Luxus bei ihrem eigenen Webauftritt nicht missen.

Es gibt zahlreiche verschiedene CMS-Systeme, wobei jedes seine Vor- und Nachteile mit sich bringt. Joomla gehört zu den wichtigsten und auch meist verbreitetsten CMS-Systemen im Open-Source-Bereich. Da das Medium, auf dem das CMS Joomla dargestellt und verwaltet wird, das World Wide Web ist, gehört Joomla zu der Gruppe der webbasierten Content-Management-Systeme. Man bezeichnet diese auch als WCMS.

Joomla findet seinen Ursprung in Australien. Im Jahr 2000 begann die australische Software-Firma „Miro International“ mit der Entwicklung des Mambo-CMS und veröffentlichte den Quellcode fast unverzüglich als Open Source. Daraus entstand eine Entwicklergemeinde, die das Projekt vorantrieb. Das Mambo-Projekt wurde kurze Zeit später aufgespalten und es entstand neben der Open-Source-Software auch eine kommerzielle Version namens Mambo 2002, die später in Jango umbenannt wurde. Joomla wurde im Jahr 2005 ins Leben gerufen und scheint daher ein sehr junges System zu sein, was so allerdings nicht stimmt. Tatsächlich basiert es aber auf dem Open-Source-Web-Content-Management-System Mambo und stellt eine eigene Weiterführung dieser Plattform dar.

Die prämierte Software wird nun unter der GNU-General-Public-Licence auf einer eigenen Schiene weiterentwickelt. Um die Idee einer Entwicklergemeinschaft zu betonen, entschied man sich für den Namen Joomla!. Es handelt sich dabei um

eine Lautumschreibung des Suaheli-Wortes »jumla« und kann mit »als Ganzes« übersetzt werden. Die aktuellste Version von Joomla ist die Version 1.6, welche viele Neuerungen im Bereich der Verwaltung speziell im Backend-Bereich beinhaltet.

Es gibt neben dem Content-Management-System Joomla natürlich noch unzählige andere ähnliche Systeme. Dazu gehören Drupal, Wordpress, vBulletin und Blogger.

Wordpress ist das wohl bekannteste CMS-System. WordPress begann als reines Blog-Script, wandelte sich aber im Laufe der Zeit zu einem echten CMS-System. Dank einer großen Entwicklergemeinde, die das CMS-System immer weiterentwickelt und Plugins geschrieben hat und, dank Webdesignern, die massenhaft sogenannte WordPress-Themes produzieren, lässt dieses CMS keine Wünsche offen.

	Im Einsatz	Marktanteil
1. Wordpress	14,9 %	54,4 %
2. Joomla	2,7 %	9,9 %
3. Drupal	1,7 %	6,2%
4. vBulletin	1,4 %	5,1%
5. Blogger	0,8 %	0,8%

Neben den bekannten CMS-Systemen, welche eine Datenbank benötigen, wird die Anzahl von Content-Management-Systemen, die keine Datenbank benötigen, immer häufiger. GetSimple und SkyBlueCanvas sind zwei nennenswerte Vertreter dieses Segments.

Einen etwas anderen Ansatz ist man bei „Pulse CMS“ gegangen: Das rund 844 KB große Redaktionssystem, das aus knapp 100 Dateien besteht, ist für Webmaster gedacht, die unkompliziert editierbare Bereiche in PHP-Dateien definieren möchten. Die sogenannten „Blöcke“ können durch Pulse-CMS definiert und mit einem komfortablen WYSIWYG-Editor bearbeitet werden.

Der Grundgedanke besteht also darin: Per Backend lassen sich dynamische Blöcke erstellen, die dann auf der fertigen Website per include eingebettet werden.

Dies setzt natürlich voraus, dass die Website mit PHP entwickelt wurde und einer bestimmten Struktur entspricht. Dennoch sieht das kleine CMS interessant aus und dürfte der Notwendigkeit von kleinen Projekten gerecht werden, Inhalte einfach editieren zu können.

Der mitgelieferte Editor zum Erstellen von Inhalten sieht vielversprechend aus und bietet schöne Features. So kann der Editor-Bereich beispielsweise in den "Vollbild-Modus" geschaltet werden. Ein Minus stellt das Einfügen von Bildern dar: Hier zeigt sich das Manko, wie es in vielen integrierten WYSIWYG-Editoren der Fall ist. So können Bilder nur eingefügt werden, wenn der komplette Server-Pfad bekannt ist. Dafür bietet Pulse-CMS einen guten Image Browser, über den sich auch Bilder hochladen lassen. Auf Wunsch kann sogar eine fertige Bildergalerie generiert werden. Pulse CMS dürfte für manche Zwecke sicherlich eine interessante Alternative sein. Die Code-Bausteine für die include-Funktion von PHP werden zu jedem Block dynamisch generiert, so dass ein Projekt sehr schnell mit der Bearbeitungsfunktion ausgestattet werden kann. Die geringe Dateigröße des kompakten Content-Management-Systems spricht für sich.

Die Auswahl an Redaktionssystemen ist mittlerweile immens und es steht bei vielen Projekten schon an der Tagesordnung, die Website über ein CMS pflegbar zu machen. Die Thematik des Content-Managements ist sehr komplex und es gibt daher etliche Faktoren, welche die Wahl eines geeigneten Systems beeinträchtigen. Nicht zuletzt hängt es vom Projekt und der Agentur (oder dem Freelancer) ab, mit welchem CMS gearbeitet wird. Neben der Flexibilität eines Systems, ist die Einarbeitungszeit zu beachten, die mit einer neuen Anwendung verbunden ist. Oft werden auch individuelle CMS-Systeme entwickelt, um dem Anspruch eines Projekts gerecht zu werden. Ein CMS sollte so gewählt werden, dass es auch für größere Projekte verwendet werden kann. Skalierbarkeit, Anpassbarkeit und Sicherheit sind wichtige Aspekte, die bei der Wahl im Vordergrund stehen sollten. Neuigkeiten aus der Welt der Inhaltsverwaltung gibt es bei www.Contentmanager.de. Technische Details, Demos und weitere Infos sollte man sich bei Seiten wie www.opensourceCMS.de oder www.cmsmatrix.org holen.

Ein weiterer Punkt ist die Benutzerfreundlichkeit und Funktionalität eines Content-Management-Systems, denn schließlich muss das System von Entwicklern und Redakteuren bedient werden können.

2 Vorbetrachtung

Nach einer kurzen Einführung in das Content-Management-System Joomla! 1.5 und den damit verbundenen Begrifflichkeiten sowie Vorrausetzungen für den Umgang mit Joomla, wird die Aufgabenstellung sowie die Zielsetzung eruiert.

2.1 Joomla! 1.5 – Einführung

2.1.1 Begriffserklärung

Frontend und Backend

Diese beiden Begrifflichkeiten spielen in der Welt der Content-Management-Systeme eine elementare Rolle. Das Frontend ist die Webseite, wie sie nach außen hin dargestellt wird bzw. wie sie der Besucher oder der dort angemeldete Benutzer sehen. Das Backend hingegen beschreibt den Bereich eines Content-Management-Systems, welcher für die Verwaltung der Webseite zuständig ist. Im Falle eines webbasierten Content-Management-System wird dieser Bereich ebenfalls in dem Browser angezeigt. Für das Backend dürfen ausschließlich berechtigte Personen Zugriff erhalten, um dort Konfigurationen vorzunehmen oder neue Inhalte zu erstellen. Das Backend findet man unter einer anderen URL als die der eigentlichen Webseite.

Zugriffsrechte

In einem webbasierten Content-Management-System werden Benutzer – oder Gruppennamen vergeben und diese mit unterschiedlichen Zugriffsrechten versehen. Dies kann zum Beispiel ein einfacher registrierter Benutzer sein oder ein Auto oder ein Editor bis hin zum Superadministrator, der vollen Zugriff auf alle Bereiche des Backends besitzt. Abhängig von den Rechten erscheint das Frontend dann mit anderen Inhalten oder der Möglichkeit, Inhalte direkt auf dem Frontend zu bearbeiten, oder aber der Benutzer erhält die Erlaubnis, im Backend zu arbeiten.

Inhalte

Dieser Begriff bildet den wichtigsten Bestandteil eines Content-Management-Systems. Dies können einfache Texte sein sowie Bilder, Videos, Musikdateien, Links oder eine Kombination aus allem. Um den Überblick über die Inhalte zu behalten, bettet man sie in Strukturen ein, beispielsweise Texte in verschiedene Kategorien. Auch die Kategorien sind natürlich Inhalte, die verwaltet werden müssen. [1]

Extensions

Extensions (Erweiterungen) ist der Obergbegriff für Plugins, Komponenten, Module und Templates. Diese Komponenten dienen der funktionalen Erweiterung der Joomla Umgebung und somit auch der Webseite. Gerade bei neueren Technologien, wie zum Beispiel jQuery, ist man auf solche Extensions angewiesen.

Plugins

Joomla!-Plugins dienen unterschiedlichen Zwecken. Sie werden erstellt, um die auszugebenen Daten zu manipulieren oder auch zusätzliche Funktionalitäten für Joomla zu erstellen. Es gibt viele verschiedene Typen von Plugins, wie zum Beispiel Search, User, System, Editor, Content oder Authentication.

Komponenten

Ebenso wie Plugins stellen Komponenten neue Funktionalitäten für Joomla bereit. Die Komponenten haben aber allerdings einen eigenen Bereich in der Joomla Administration. Beispiele für Komponenten können sein: Online-Shop, Bildergalerie oder auch Forensysteme.

---

[1] Vgl. Hagen Graf: joomla-1.5 URL: http://cocoate.com/de/joomla-15/begriffe-konzepte-und-ueberlegungen/struktur-eines-web-content-management-systems-wcms

Module

Module werden am häufigsten in Verbindung mit den Komponenten genutzt, um zum Beispiel Inhalte von Komponenten auf der Webseite oder einen bestimmten Bereich der Webseite anzuzeigen. Das Modul „1. Bild“ zum Beispiel, zeigt das erste Bild aus der Komponente „Bildergalerie“.

Templates

Ein Template definiert das komplette Erscheinungsbild einer Webseite in dem System Joomla. Sie besteht aus mindestens 2 Dateien. Einer HTML-Datei für die Struktur der Seite und einer CSS-Datei für das Design.

API

Die API stellt das Bindeglied zwischen dem Joomla!-Kern und dem Entwickler.

Workflow

Dem Begriff Workflow begegnet man heutzutage in jedem Bereich des Lebens. Hierbei handelt es sich um eine Art Anleitung ein Produkt zu schaffen oder einen Arbeitsablauf zu steuern bzw. zu definieren. Dieser Begriff spielt in dem Bereich CMS ebenso eine große Rolle, da hier ein organisierter Arbeitsablauf eine wichtige Komponente zur Realisierung von Webprojekten darstellt. Besonders wichtig ist ein guter Workflow, wenn mehrere Personen an Inhalten bzw. an der Konfiguration einer Webseite arbeiten.

Konfigurationseinstellungen

Hinter diesem Begriff verbirgt sich eine Vielzahl von Einstellungsmöglichkeiten für die Seite des CMS. Beispiele hierfür sind: Titeltext im Browserfenster, Schlüsselwörter für Suchmaschinen, Farbeinstellungen, Schalter, mit denen Benutzern Zugriffsrechte erteilt oder genommen werden können, bis hin zu der Möglichkeit die Seite online oder offline zu setzen.

Barrierefreiheit

Dieser Begriff hat in den letzten Jahren zunehmend an Bedeutung gewonnen. Mit Barrierefreiheit verbindet man die Einhaltung von Webstandards und die strikte Trennung von Inhalten, wie Bildern oder Text, und dem Layout, welches durch den Einsatz von Cascading-Style-Sheets erzeugt werden soll. Dies betrifft allerdings nicht nur das Frontend. Zunehmend soll der Administrationsbereich auch barrierefrei werden. Das W3C hat Standards dazu aufgestellt.

XML

Prinzipiell kann XML für jede Art von Datenbeschreibung, -speicherung und -austausch genutzt werden. Die Vorteile sind die grosse Verbreitung und damit der geringe Lernaufwand, die leichte Lesbarkeit für Menschen und Maschinen und die Portabilität. Gegenüber einem eigenen kompakten *Binärformat* sind die Nachteile der grössere Speicherbedarf und u.U. langsamere Verarbeitung. Beides ist heutzutage in den meisten Fällen nicht mehr wichtig, so dass die Vorteile die Nachteile überwiegen. Entsprechend hat sich XML in vielen Bereichen durchgesetzt.

2.1.2 Der Aufbau eines CMS- Systems

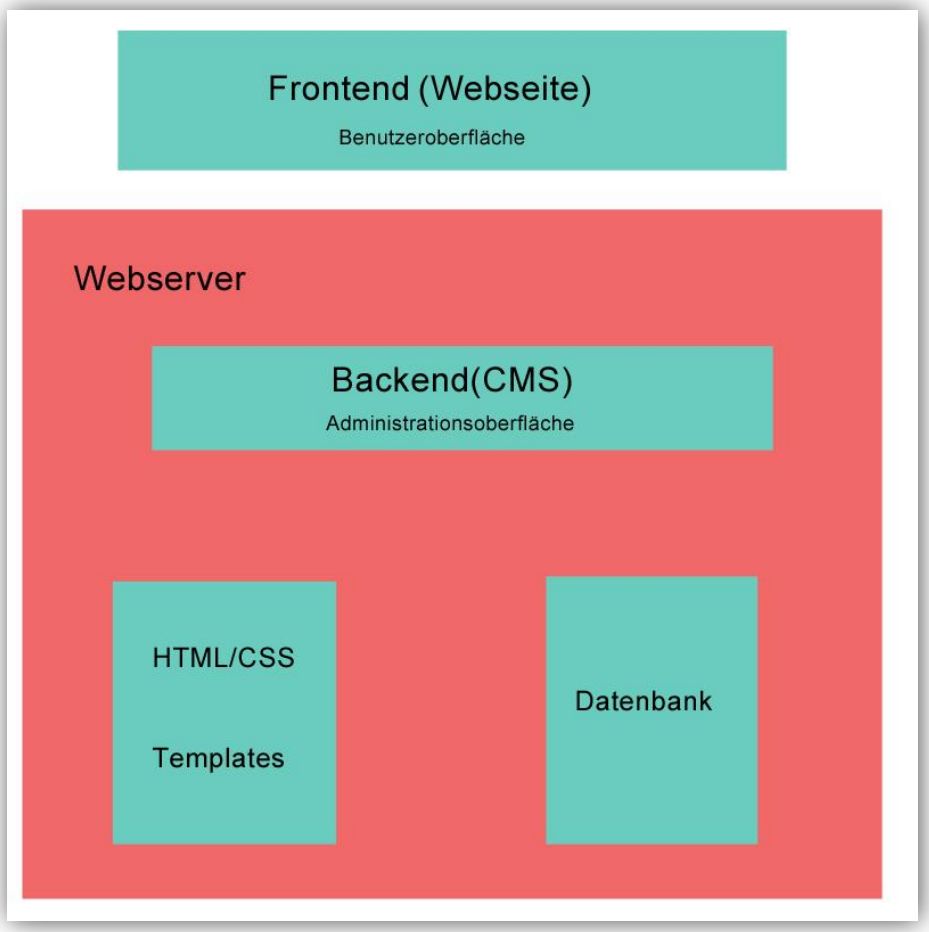


Abbildung 1: Aufbau CMS

2.2 Aufgabenstellung

Die Studenten der Multimediatechnik erhalten im Laufe ihres Studiums einen theoretischen Einblick in die Technologie des Content-Management-Systems Joomla!. Vor diesem Hintergrund soll im Rahmen dieser Diplomarbeit auf Basis des Joomla!-CMS ein Praktikum entwickelt werden, welches dazu dienen soll, Studenten im Bereich Multimediatechnik im 2. Semester an dieses CMS heranzuführen.

Ziel ist die Realisierung eines Komplexpraktikums zur Erzeugung und Integration von Templates in ein CMS.

Die Durchführung eines Praktikums kann vor allem daran scheitern, dass sich im Verlauf der Arbeit herausstellt, dass verschiedene Faktoren, wie Zeit oder Wissensumfang der Studenten, nicht mit dem des Praktikums einhergehen.
Daher soll das Praktikum in Form eines Tutorials aufgestellt werden. Es wird eine ausführliche Praktikumsanleitung, mit detailliertem und vor allem anschaulichem Inhalt erstellt werden. Hinzu kommt, dass der Praktikumsleiter die einzelnen Schritte mit den Studenten zusammen abarbeitet, um eventuelle Fragen schnell beantworten zu können. Das Praktikum soll von jedem Studenten allein durchgeführt werden. Von einer Gruppenarbeit, bzw. einer Aufteilung der einzelnen Gebiete auf mehrere Studenten ist abzuraten, da die einzelnen Bereiche des Praktikums durchaus ineinander übergreifen können, bzw. sogar verschachtelt sein können.
Das Praktikum sollte sich über 3 oder 4 Einheiten erstrecken. Der Inhalt dieses Praktikums soll sein, dass die Studenten ein eigenes Template in Joomla! 1.5 erstellen sollen. Dieses Template soll auf einem HTML/CSS-basierten Layout entstehen. Es soll von Grund auf, ohne Zuhilfenahme bestehender Joomla-Templates entstehen.

2.3 Anforderung an das Komplexpraktikum

Dem Studenten soll mit Hilfe dieses Praktikums der Umgang mit Joomla und damit verbunden HTML und CSS nahegebracht werden. Durch dieses Praktikum soll der Student für die Benutzung dieses oder auch ähnlicher Content-Management-Systeme sensibilisiert werden. Durch die Entwicklung einer Webpräsenz von Beginn an ohne jegliche Vorlagen, erlangen die Studenten außerdem viel zusätzliches Wissen im Bereich CSS und HTML. Hierbei stehen besonders Techniken im Fokus, die auch in anderen Bereichen der Webprogrammierung relevant sind.

Außerdem soll den Studenten ein strukturierter Arbeitsablauf bei der Erstellung einer Webseite dargestellt werden. Aus diesem Grund soll das Praktikum so gestaltet werden, dass der Student selbstständig zum Ergebnis kommt, ohne den Praktikumsleiter um Hilfe zu bitten. Dafür muss allerdings beachtet werden, dass auf bestimmte Bereiche, wie zum Beispiel PHP oder Javascript, auf Grund des Wissensstatus der Studenten im 2. Semester, nicht eingegangen werden kann. Etwaige Syntax wird den Studenten vorgegeben und kurz deren Funktion benannt.

Um die Motivation der Studenten zu wecken, sich nach der erfolgreichen Durchführung des Komplexpraktikums, zusätzliches Wissen über den Gebrauch von Joomla anzueignen, sollen bei dem Praktikum für die Studenten moderne und interessante Themen bearbeitet werden. Außerdem soll das komplette Screendesign des zu erstellenden Templates in Form einer Photoshopdatei den Studenten zu Verfügung gestellt werden, was zusätzlich als Motivation dienen soll, sich an eigene Projekte neben dem Studium zu wagen.

Da bereits neuere Versionen von Joomla vorhanden sind, wird in dem Praktikum auf die etwaigen größeren Änderungen und Unterschiede der Versionen eingegangen.

3 Entwurf

Bevor es zur Erstellung des Komplexpraktikums kommt, wird eine Entscheidung über den Umfang, die Struktur und die technische Umsetzung getroffen. Außerdem muss das Praktikum in inhaltlich sinnvoll zusammenhängende Blöcke eingeteilt werden.

3.1 Themenwahl für das Komplexpraktikum

Der Bereich Content-Management-Systeme bietet eine Vielzahl verschiedener Themen, welche für ein Komplexpraktikum für Multimediastudenten in Betracht kommen. Allerdings wird die Auswahl durch die bis dahin von den Studenten erlangten Fertigkeiten im Bereich der Webprogrammierung eingeschränkt. Ein Praktikum, was ausschließlich der Bedienung des Backend von Joomla abzielt, stellt keinen wirklich praktischen Nutzen dar, welchen ein solches Praktikum aber besitzen sollte. Vorliegend wurde der Bereich der Template Entwicklung ausgewählt. Es soll sich dabei um ein Template auf Basis eines HTML/CSS-Layout handeln. Es soll als Motivation dienen, nach dem Praktikum eigenständig eine Webseite erstellen zu können und dies zudem noch auf Basis eines Content-Management-Systems.

Das Template soll eine Portfolio Seite eines Webdesigners darstellen. Auf dieser Grundlage soll es den Studenten möglich sein, ihre eigenen Vorstellungen von einer Webseite umzusetzen.

3.2 Konzeptionierung des Templates

Anfangs stellt sich die Frage, welchen Funktionsumfang das Template haben sollte. Da es in dem Zeitraum des Praktikums nicht möglich ist, den kompletten Funktionsumfang des Joomla!-Content-Management-Systems zu erläutern und anzuwenden, muss sich das Template dem Komplexpraktikum unterordnen. Allerdings sollte es einen gewissen Anlass geben, so dass die Motivation, daran weiter zu arbeiten, gegeben ist.

3.2.1 Auswahl des Layout zum Joomla Template

Um ein geeignetes Layout für das Template zu finden, wird zunächst geprüft, welche Funktion das fertige Template haben soll. Wie in 3.1 erwähnt, soll dem Funktionsumfang dem einer Portfolioseite gleichen. Mit dieser Überlegung ist es nun möglich, den Umfang der Navigation und die damit verbundene Anzahl an Unterseiten festzulegen. Daher legen wir die Anzahl der Menüpunkte auf maximal 6 fest.

Da nun der grobe Umfang des Templates definiert wurde, kann man die Wahl eines geeigneten Layouts einengen. Damit fällt die Entscheidung auf ein 2-Spalten-Layout. Durch ein Layout dieser Art, ist eine Trennung von Navigation und Inhalt garantiert. Die Navigation ist in dem Bereich *Sidebar* vorgesehen. Es wird sich hierbei um eine vertikal ausgerichtete Menüführung handeln. In dem *Header*-Bereich soll keine Interaktion in Form eines Menüs, mit dem Benutzer stattfinden. Dieser soll lediglich ein Logo bzw. einen Seitenname beinhalten, sowie Social-Network-Icons, welche verlinkt werden können. In dem Bereich *Content* wird der komplette Inhalt der Seite dargestellt. Er soll ausreichend Platz bieten, um verschiedene Arten von Inhalten wie Text, Bilder oder Videos gut darstellen zu können.

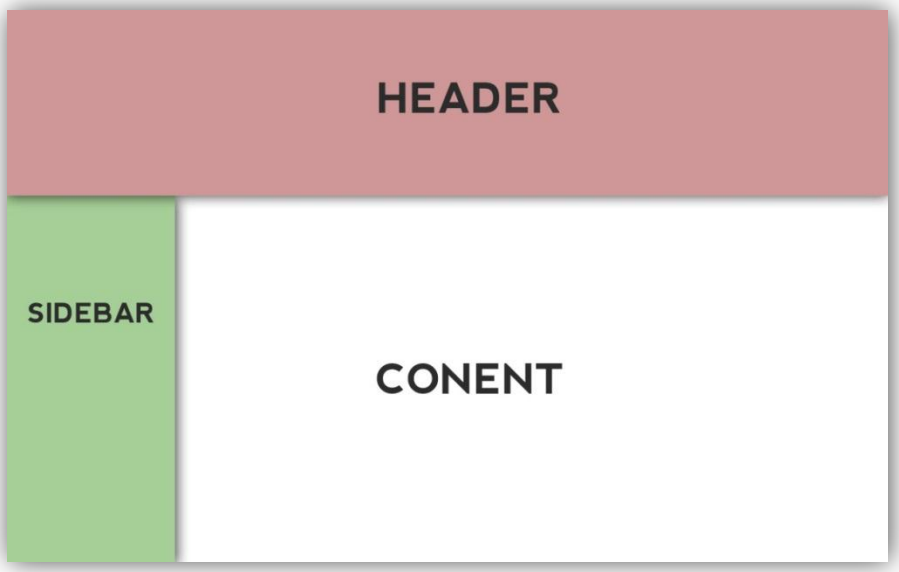


Abbildung 2: CSS layout

3.2.2 Das Screendesign zum Joomla-Template

Die Screendesign-Variante, welche als Grundlage für das Joomla-Template genutzt werden soll, hat den Anspruch ein ansprechendes, modernes und zugleich übersichtliches Aussehen zu erhalten. Für die Erstellung von Screendesigns sind vielerlei Vorüberlegungen zu tätigen, bevor es zur eigentlichen Umsetzung dieser kommt. Zu allererst wird festgelegt, mit welchen Werkzeugen das Screendesign erstellt werden soll. Hierfür bieten das Grafikprogramm Photoshop sowie Illustrator von Adobe die besten Voraussetzungen, um ein Screendesign nach seinen eigenen Ansprüchen zu erstellen. Eine weitere grundlegende Überlegung im Hinblick auf die Erstellung des Designs, ist die Wahl der Bildgröße, in der das Layout erstellt werden soll. Hierbei gibt es allerdings keine verbindlichen Aussagen, welche Auflösung am geeignetsten erscheint. Allerdings liegen Webstatistiken vor, welche Aufschluss über die Verbreitung der verschieden Auflösungen geben können.

1024 x 768	**36.8%**
1280 x 1024	**26.9%**
1680 x 1050	**6.9%**
1280 x 800	**4.2%**
1920 x 1200	**3.7%**
1440 x 900	**3.5%**
1152 x 864	**3.4%**
1600 x 1200	**2.8%**
800 x 600	**2.5%**
1400 x 1050	**1.7%**
1280 x 960	**1.2%**
1280 x 768	**0.9%**
Andere	**5.5%**

Da es heutzutage fast schon zur Normalität gehört, den Inhalt einer Webseite in maximal 960 Pixeln Breite darzustellen, bietet die Bildgröße von 1280x1024 Pixeln die Grundlage für das Screendesign des Joomla-Templates. Um nun die gerade erwähnten 960 Pixel Breite nicht zu überschreiten, werden Hilfslinien in Photoshop verwendet.

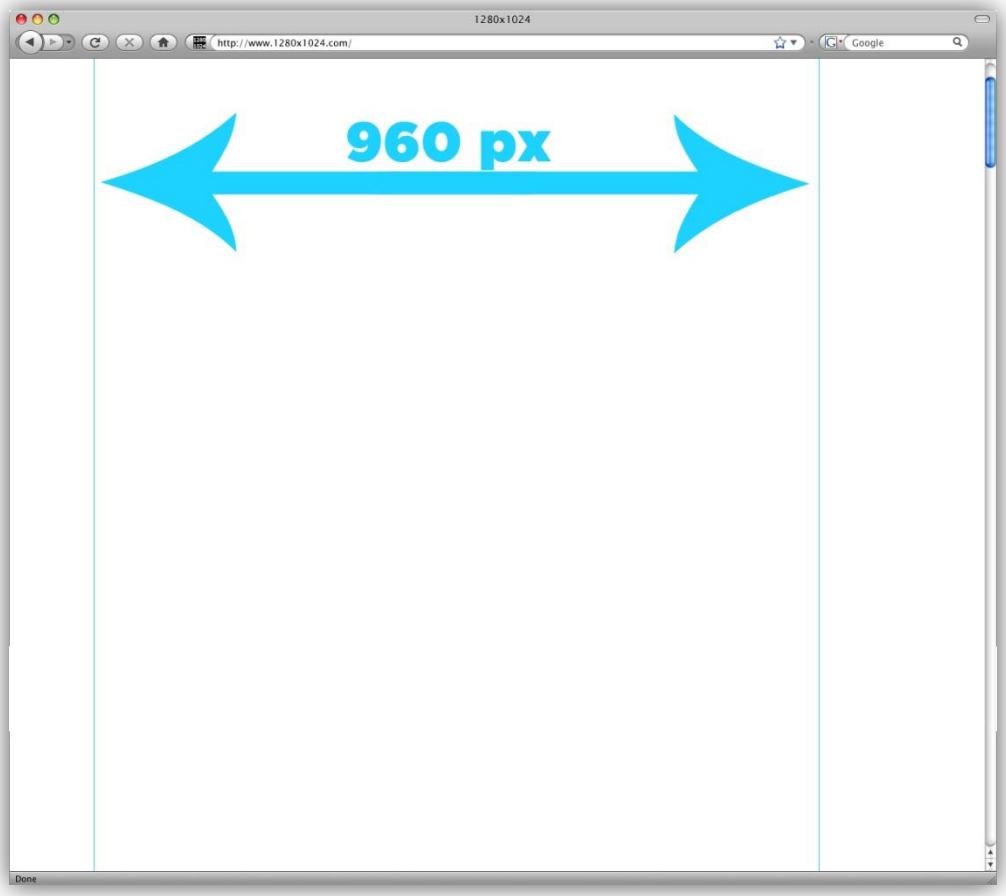


Abbildung 3: wrapper-breite

Man kann selbstverständlich Hilfslinien auch für den Head-Bereich, sowie für alle anderen Bereiche der Seite verwenden. Somit wird es dem Programmierer später leicht gemacht, genaue Pixelwerte in das CSS zu übertragen. Außerdem kann man somit eventuelle Bereiche aus dem Screendesign, welche man als Grafik in die Seite einbauen möchte, viel leichter freistellen.

Auf Grundlage dieser Vorüberlegungen wird ein Screendesign für das Joomla-Template angefertigt.

Folgendes Screendesign soll für das Joomla-Template verwendet werden:

Abbildung 4: screendesign

3.2.3 Analyse vom Screendesign zum Webdesign/Template

Background

Das Screendesign gibt eine Hintergrundgrafik für das Joomla!-Template vor. Diese besteht aus einer PNG-Grafik, welche später mittels eines CSS Befehls horizontal wiederholt werden soll. Eine Hintergrundgrafik einzubauen, ist an sich nichts besonders schwieriges. Es kommt allerdings darauf an, wie genau diese Hintergrundgrafik eingebettet wird und wie sie sich im Kontext funktionaller Aspekte verhält. Für das Joomla!-Template ergeben sich daraus folgende Überlegungen: Hat unser Template eine flexible Breite? Falls ja, muss die Hintergrundgrafik aufwändiger integriert werden, als wenn es eine feste Breite hat. Bei einer festen Breite kann die Hintergundgrafik, welche als GIF oder PNG vorliegen wird, einfach per CSS in das umgebende DIV legen und zentrieren. Bei einer flexiblen Template Breite könnte man das auch machen, allerdings verschwindet die Hintergrundgrafik dann irgendwann hinter dem Content, wenn die Breite erhöht wird.

Wrapper-Background

Auch dieser Bereich des Templates wird von dem Screendesign vorgegeben. Ähnlich wie bei dem oben genannten Hintergrund der Seite besteht dieser aus einer PNG-Grafik, welcher allerdings vertikal wiederholt werden soll. Der Schatten, den der Wrapper auf den Seiten-Hintergrund wirft, kommt ebenso von dieser PNG-Grafik.

Header

Alle Grafiken in dem Head Bereich, liegen über dem Wrapper-Background. Sie bestehen aus separaten Grafiken, da diese später wie zum Beispiel bei den Social-Network-Icons als Butten genutzt werden sollen. Die Logo Grafik sollte unabhängig von der Textgröße des Logos, die selbe Höhe haben wie der Head-Bereich. Somit müssen im CSS keine überflüssigen Positionierungsangaben gemacht werden, was auch zu Problemen im Bereich Browserkombatibilität führen kann.

Navigation

Die Hauptnavigation oben befindet sich in der linken Seite innerhalb des Wrappers. Da die Navigation aus einem Modul bestehen soll, besteht die Möglichkeit für den Benutzer des Templates, nach Belieben Menüeinträge hinzuzufügen oder zu entfernen. Aus diesem Grund wurde eine vertikale Menüanordnung gewählt, damit für etwaige Änderungen ausreichend Platz dem Nutzer des Templates zu Verfügung steht. Die aktiven Menüpunkte sollen mit einem weißen Balken hinterlegt werden. Hierfür soll auf eine Grafik zur Darstellung verzichtet werden, weil eine Realisierung durch CSS möglich ist und somit kein Performance Verlust eintritt.

Footer

Der Footer schließt mit dem Content ab und besitzt die gleiche Breite wie der Wrapper. Was passiert, wenn man im Content nur einen Satz stehen hat und dadurch die Content Höhe extrem gering wird? Wie sieht es dann unter dem Footer aus? Geht es dann einfach weiter oder verlaufen die beiden Grauschattierungen weiter? Auf diese Fragen wird in der Implementierung des Templates später eingegangen.

Damit wurden die ersten kritischen Stellen dieses Screendesigns identifiziert und mögliche Lösungswege beschrieben. Auf dem Weg der Umsetzung werden sicher noch einige Herausforderungen auftauchen und es wird notwendig sein, neue, nicht bedachte Features für das Template einzubauen.

3.2.4 Technische Vorrausetzungen für das CMS-Joomla!

Für die Installation eines von Joomla ist ein funktionierender Webserver erforderlich. Hierbei besteht die Möglichkeit einen Apache-Server ab Version 1.13.19 oder einen Microsoft IIS zu installieren. Außerdem muss die Scriptsprache PHP ab Version 4.3 enthalten sein. Die Unterstützung für MySQL und Zlib muss in PHP kompiliert sein. Das Datenbanksystem MySQL muss ab Version 3.23.x bereit stehen. All diese Komponenten sind für eine erfolgreiche Joomla!-Installation erforderlich.

3.2.5 Das Template-Gerüst

Das Template soll in die Joomla!-CMS-Version 1.5 implementiert werden. Wie im vorangegangen Kapitel beschrieben, soll es auf der Grundlage eines Screendesigns enstehen. Dafür soll ein Grundgerüst, bestehend aus HTML und CSS, verwendet werden. Das Hauptaugenmerk soll dabei auf die Übersichtlichkeit gelegt werden. Dadurch wird gewährleistet, dass es stets möglich ist, schnell das HTML/CSS -Konstrukt zu verstehen und Anpassungen vorzunehmen. Besonders gilt dies für das Stylesheet, welches mit klar beschriebenen IDs und Klassen bestückt werden soll.

index.php

Die index.php stellt das eigentliche Template dar. Diese besteht aus XHTML, PHP und Joomla Anweisungen. Da der Joomla-Core auf XHTML ausgerichtet ist, sollte auch ein entsprechender DOCTYPE verwendet werden. Im Grunde sind die HTML-Tags im Vergleich zu einer `index.html` die gleichen. Der einzige Unterschied zu einer `index.html` besteht darin, dass Inhalte dynamisch mittels PHP- Anweisungen erzeugt werden. Außerdem werden in den Bereichen, in welchen dynamische Inhalte erzeugt werden sollen, jdoc-Tags eingefügt.

Codebeispiel eines jdoc-Tag

```
<div id=”content”>
<jdoc:include type="modules" name="content" style="xhtml" />
</div>
```

In dem oben abgebildeten Codebeispiel für ein jdoc-Tag stellt der Parameter `type` den Typ des Einschubs dar, in diesem Fall `modules`. Die Position des Moduls (top, right, left…) auf der Webseite, wird von dem Parameter `name` gekennzeichnet.

template.css

Die Datei `template.css` wird in den css-Ordner angelegt und wird wie üblicherweise mit css-Befehlen gefüllt. Hiebei ist zu beachten, dass dieses css-File auch den Namen `template.css` erhält, da Joomla! sonst nicht auf diese Datei zugreifen kann. Die Gestaltung der Datei ist nicht vorgegeben, es gibt jedoch Standardbezeichnungen für verschiedene CSS-Elemente. Ein Beispiel hierfür ist das Style-Tag `contentheading{}`, welches von Joomla! erzeugt wird und in der template.css angepasst werden kann.

templateDetails.xml

Hinter dieser xml-Datei verbirgt sich die Bauanleitung für den Template Installer. Hier werden alle für die Installation benötigten Meta-Daten beschrieben. Wenn das Template über den Joomla!-Installer installiert wird, liest PHP die `templateDetails.xml` aus und kopiert die Dateien an den Platz, der in der XML-Datei angegeben worden ist. Die Datei wird in dem root-Verzeichnis des Templates angelegt.

Codebeispiel für den Aufbau einer templateDetails.xml

```
<install version="1.5" type="template">
	<name>Name der Seite</name>
	<version>1.0</version>
	<creationDate>11. Mai 2012</creationDate>
	<author>Name des Autors</author>
	<authorEmail>Autor@mail.de</authorEmail>
	<authorUrl>http://www.autor.de</authorUrl>
	<copyright></copyright>
	<license>GNU/GPL</license>
	<description>Templatebeschreibung</description>
	<files>
		<filename>css/template.css</filename>
		<filename>index.php</filename>
		<filename>templateDetails.xml</filename>
<filename>teplate_thumbnail.png</filename>
	</files>
	<positions>
		<position>head</position>
		<position>left</position>
<position>right</position>
		<position>content</position>
		<position>search</position>
		<position>footer</position>
	</positions>

</install>
```

3.3 Konzeptionierung des Komplexpraktikums

Durch die vorangegangen Abschnitte wird der Umfang des Projekts, also der Erstellung eines Templates für das CMS Joomla!, deutlich. Deshalb muss das Praktikum auf verschiedenste Aspekte hin diskutiert werden, um einen Überblick davon zu erhalten, wie viel von der Entwicklung eines Templates schlussendlich von den Studenten erarbeitet werden soll. Außerdem ist die Entscheidung über den Zeitrahmen, der für die Durchführung dieses Komplexpraktikums erforderlich ist, sehr wichtig und muss untersucht werden.

3.3.1 Die Grundstruktur des Komplexpraktikums

Das Komplexpraktikum wird in 4 inhaltlich zusammenhängende Praktika unterteilt. Zu Beginn, also in der ersten Einheit, sollte eine klare Definition über die Ziele des Praktikums stattfinden. Dies geschieht in Form einer Einleitung, welche sich in den Praktikumsunterlagen der Studenten befindet. Dadurch wird den Studenten ermöglicht, sich ein Bild davon zu verschaffen, was das eigentliche Ziel des Praktikums ist. Die Form der Gruppenarbeit, welche mit Sicherheit für viele Bereiche im Laufe des Studiums Vorteile mit sich bringen kann, ist für dieses Praktikum allerdings keine gute Lösung. Durch das schrittweise Abarbeiten der einzelnen Aufgabengebiete, würde sich eine Gruppenarbeit negativ auf die Arbeit des einzelnen Studenten auswirken.

Das Protokoll

Für jede der 3 Praktikumseinheiten soll den Studenten ein Protokoll bzw. eine Praktikumsanleitung vorgelegt werden. Hierbei ist es wichtig bei jedem der einzelnen Protokolle mittels eines Deckblattes eine klare Beschreibung des Themenbereiches des jeweiligen Praktikums zu formulieren. Des Weiteren sollte noch einmal zu Beginn in diesem Protokoll darauf eingegangen werden, welcher der aktuelle Stand des Komplexpraktikums ist. Dies bietet den Studenten die Möglichkeit, sich über den gegenwärtigen Zustand des Praktikums zu informieren und eventuelle Rückstände oder Probleme zu lösen.

Die Praktikumsprotokolle sollen in einer Schritt-für-Schritt-Anweisung angefertigt werden. Dabei sollte beachtet werden, möglichst sämtliche Punkte, die für die erfolgreiche Umsetzung des Praktikums erforderlich sind, zu erfassen. Da der Student im ungünstigsten Falle keine Fähigkeiten im Umgang mit einem Content-Management-System besitzt, muss er besonders im ersten Praktikum für die Arbeit mit einem CMS-System sensibilisiert werden.

Außerdem kann nicht vorausgesetzt werden, dass der Umgang mit HTML und CSS von den Studenten beherrscht wird. Auch wenn es sich um Studenten der Multimediatechnik handelt, ist dies keineswegs trivial. Allerdings soll das Praktikum keine theoretischen Teile, zum Beispiel zum Thema Cascading-Style-Sheets, beinhalten. Dies soll allerdings nicht ausschließen, dass einzelne theoretische Elemente vorkommen sollen, aber nur in einem Umfang, der nicht zur Unruhe oder gar zur Langeweile führen kann.

Das gestalterische Element spielt für das Verständnis, sowie für die erfolgreiche Durchführung des Komplexpraktikums eine tragende Rolle. Durch verschiedene grafische Elemente bezugnehmend auf Textinhalte, soll den Studenten angezeigt werden, welche Teile des Praktikums theoretischer Natur sind und welche Teile der praktischen Durchführung durch den Studenten bedürfen. Zusätzlich wird der Quellcode, welcher für die erfolgreiche Durchführung des Komplexpraktikums erforderlich ist, grafisch hervorgehoben. Ein grafisches Element, welches auf Hinweise, die von großer Wichtigkeit für den erfolgreichen Ablauf der Praktika aufmerksam macht, soll ebenso vorhanden sein. Zusätzliche Hinweise, welche eine bestimmte Aktion des Studenten näher beschreiben, sind auch vorgesehen.

*Grafische Darstellung für den **Quellcode***

```
body {
    background:url(../images/back.png) repeat;
    }
```

*Grafische Darstellung für den **wichtige Hinweise***

Ein weiterer wichtiger Aspekt in dem Bereich der Gestaltung des Praktikumsprotokolls, ist die klare Darstellung der Schrittanweisungen. Diese sollen mittels eines Symbols, welches dem Studenten eine Aktion vermitteln soll, dargestellt werden. Dabei ist außerdem noch zu beachten, möglichst jede Aktion, welche von den Studenten durchgeführt werden soll, mit einem solchen Symbol zu versehen. Dadurch soll vermieden werden, dass wichtige Anweisungen durch zu lange Sätze überflogen oder gar ausgelassen werden.

Symbol für die Darstellung von durchzuführenden Aktionen.

Um eine Abgrenzung zwischen den unterschiedlichen Abschnitten zu erhalten, werden zwei unterschiedliche Überschriften gestaltet

Eine Grundüberlegung, die jedoch nicht allzu großen Einfluss auf das Praktikum hat, ist das Schriftbild. Es sollen maximal 3 Schriftarten verwendet werden. Überschriften, Inhaltstext sowie Dateinamen sollen durch die verschiedenen Schriftarten abgetrennt werden. Für die Überschrift soll die Schriftart *„Cambria“*, für den Inhaltstext *„Arial“* und für die Dateinamen sowie Codebeispiele die Maschinenschrift *„Courier“* verwendet werden.

Um URLs von dem Text abzugrenzen, werden diese kursiv dargestellt. Auf verschieden Farben wird in dem Inhaltstext verzichtet. Bei den Quellcodeangaben wird allerdings farbige Schrift verwendet. Hier ist es auch von Vorteil Tags und Parameter farblich voneinander zu trennen, da man es heutzutage durch

Programme wie Dreamweaver oder Eclipse gewohnt ist. Somit wird den Studenten eine bessere Übersicht über den Code gewährt.

Überschriften werden mit Schriftgröße 18 bzw. 13 in einem Blauton erstellt. Normale Abschnitte werden stets mit Schriftgröße 14 erstellt.

4. Die Erzeugung des Komplexpraktikums

Es sollten nun alle Vorüberlegungen für die Entwicklung des Komplexpraktikums getroffen worden sein, so dass nun die eigentliche Umsetzung erfolgen kann. Hierbei wird auf die Entwicklung des Joomla!-Templates, sowie auf die Erzeugung des Praktikums wert gelegt.

4.1 Erzeugung des Joomla!-Templates

Die unter Abschnitt 3 besprochenen Planungen zur Erzeugung des Joomla!-Templates, werden nun umgesetzt. Es wird in den folgenden Kapiteln der komplette Ablauf der Erzeugung des Templates beschrieben. Die Beschreibung erfolgt durch ausgewählte Quellcode sowie Bildmaterial.

4.1.1 Lokale Joomla! Installation

Es wurde eine lokale Serverumgebung gewählt, da der Zugriff auf den Home-Bereich der Hochschule nicht garantiert werden kann und dies zu großen Zeitproblemen bei der Durchführung des Praktikums führen kann. Außerdem kann auch nicht garantiert werden, dass, wenn es zu der Durchführung des Komplexpraktikums kommt, keine Versionskonflikte mit Joomla! 1.5 und der Serverumgebung auftreten. Für die lokale Serverumgebung wird XAMPP verwendet. Dabei muss beachtet werden, dass es sich um die 1.7.3 von XAMPP handelt, da es bei neueren Versionen zu Konflikten mit Joomla! 1.5 kommen kann. Die Installationspakete, sowohl für Joomla! als auch für die XAMPP-Serverumgebung, werden später den Studenten zur Verfügung gestellt, da möglicherweise die benötigten Versionen im Internet nicht, bzw. nur umständlich zu finden sind.

Vor der Installation von Joomla! wird eine mySQL-Datenbank namens *„Joomla“* erzeugt. Hier werden sämtliche Datensätze, welche Joomla! Benötigt, abgelegt.

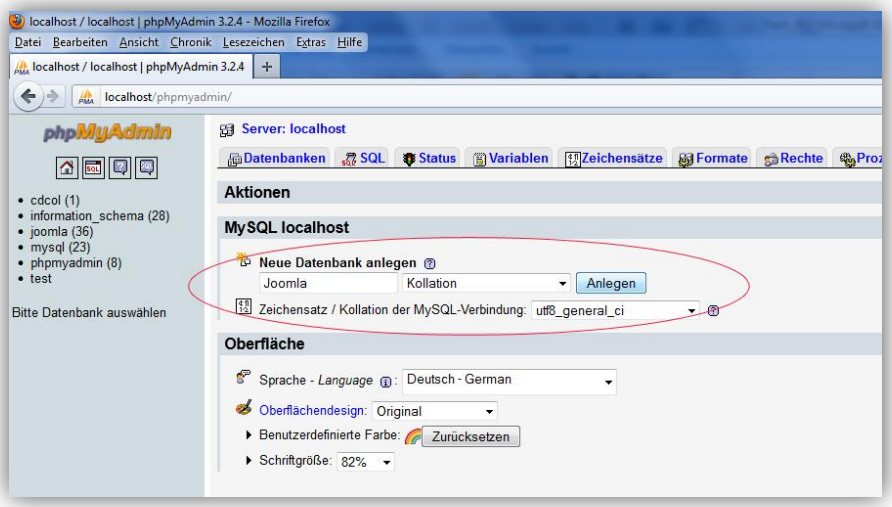


Abbildung 5: Datenbank

Bei der Installation sollen die Beispieldaten von Joomla! mit installiert werden. Diese finden zwar für das Template keine Verwendung, allerdings ist dies von Vorteil für das Praktikum, wenn es um das Verständnis für das Backend von Joomla! geht.

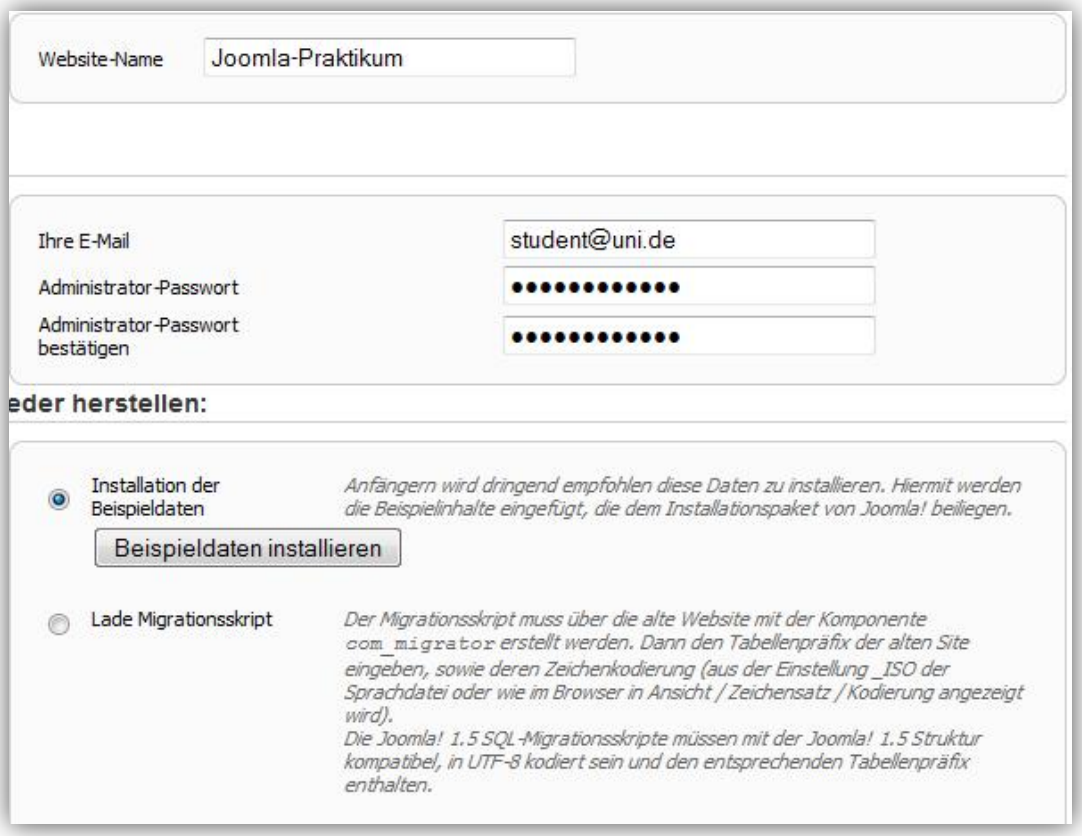


Abbildung 6: joomla Installation

Bevor es zum Abschluss der Joomla! Installation kommt, muss der Order „Installation“ aus dem Joomla!-Verzeichnis gelöscht werden. Danach gelangt man entweder auf die Startseite, welche mit den Beispieldaten, die bei der Installtion mit asgewählt worden oder man hat die Möglichkeit direkt in den Adminstrations-Bereich zu gelangen. Um Zugang zu dem Backend zu gelangen, bedarf es der Eingabe von Benutzername „root“ und dem bei der Installation vergebenen Passwort.

Um zu einem späteren Zeitpunkt zu dem Backend-Bereich zurückzukehren, bedarf es der Eingabe von localhost/Joomla/administrator. Hierbei ist zu beachten, dass „administrator“ klein geschrieben wird, da es sonst in dem Backend-Menü zu einem Fehler kommen kann.

Abbildung 7: Backend Anmeldung

4.1.2 Joomla!-Backend Konfiguration

Beispieldaten löschen

Die Beispieldaten wie Beiträge, Bereiche, Kategorien oder Menüs können gelöscht werden. Die Verwaltung der Beiträge wird über *Inhalt/Beiträge* aufgerufen. Hier werden sämtliche Beiträge in ausgewählt und in Papierkorb für Beiträge verschoben. Damit befinden sich die Beiträge in dem Papierkorb, welcher über „*Inhalt/Papierkorb: Beitrage“* aufgerufen wird. Nun werden sämtliche, sich in dem Papierkorb befindlichen Beiträge gelöscht. Die Beiträge sind die einzigen Elemente in Joomla!, man mit dieser Methode löscht. Andere Elemente von Joomla! sind nicht mit einem Papierkorb verknüpft. Nun werden alle Bereiche, Kategorien und Menüs gelöscht.

Bei den Menüs bildet der Menüpunkt „Startseite“ in dem Hautmenü eine Besonderheit, da dieser nicht gelöscht werden kann.

Module deaktivieren

Die Module, welche über *Erweiterungen/Module* erreichbar sind, werden deaktiviert.

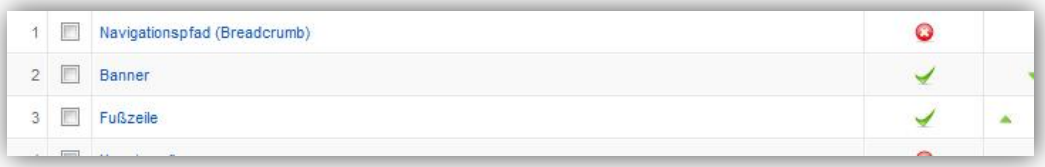


Abbildung 8: Module

Abbildung 9: Butten für aktivieren / Deaktivieren

Dabei gilt es zu beachten die Module nicht zu löschen sondern lediglich zu deaktivieren.

Lesbare URLs aktivieren

Über den Menüpunkt *Site/Konfiguration* erscheint die Hauptkonfiguration von Joomla! . Hier wird in dem Untermenü „Site“ in dem Bereich Suchmaschinen-Optimierung der Radiobutton ja für „Suchmaschinenfreundliche URLs“ ausgewählt. Somit werden die kryptischen Pfade, für den Aufruf einer Seite mit einer Vielzahl an Parametern in eine vielsagende, gut lesbare URL umgewandelt.

Abbildung 10: Seo_URLS

4.1.3 Die Dateien- und Verzeichnisstruktur für das Webprojekt

Bei der Dateien- und Verzeichnisstruktur wird auf den üblichen Aufbau für Webprojekte zurückgegriffen. Die einzige Vorgabe hierbei betrifft die css-Datei, welche den Namen `template.css` tragen muss, damit Joomla! diese später auch erkennen kann.

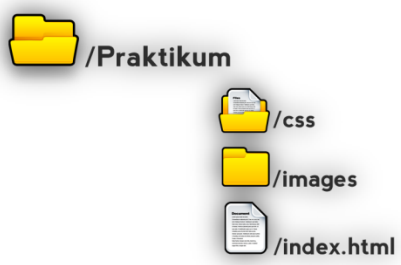


Abbildung 11: Verzeichnisstruktur

Diese Verzeichnisstruktur wird vorerst nicht in den Joomla!-Ordner implementiert, da zu Beginn nur das statische Grundlayout erstellt wird.

4.1.4 Das HTML-Gerüst

Doctype

Das W3C hat mit jeder neuen Entwicklung von HTML, XHTML oder XML die so genannte DTD (Doctype Declaration) veröffentlicht. Diese unterliegt strengen Vorschriften und ist ein fester Bestandteil von HTML, XHTML und XML. Die Doctype Deklaration teil dem Browser mit, was er anzeigen soll, wie er das CSS darstellen soll und mit welcher Sprache und welcher Art das Dokument ist.

Der Aufbau einer Document Type Definition geschieht immer nach dem selben Schema:

```
<!--Schluesselwort Argument Parameter-->
```

Es gibt eine begrenzte Anzahl an Schlüsselwörtern, die verwendet werden dürfen. Dies sind *DOCTYPE, ELEMENT, USEMAP, NOTATION, ENTITY, ATTLIST* und *SHORTREF*.

Die Wahl des DOCTYPE für die `index.html ist auf Folgenden gefallen:`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

So sollte man einen DOCTYPE wählen, bei dem alle Browser im Standard-Modus rendern und das Boxmodel von den Browsern richtig berechnet wird. Dies ist bei einem CSS-Layout von großer Bedeutung.

div-container Struktur

Da es sich bei dem Layout um ein 2-Spalten-Layout handelt, ist die Container-Struktur eher einfach gehalten. Durch die geringe Verschachtelung der einzelnen Div-Elemente ist stets ein guter Überblick geboten. An jeden der einzelnen Div-

Container wird eine ID vergeben, welche später für die Gestaltung mittels CSS unabdingbar sind. Innherhalb des <Body>-Tag wird folgende Struktur implementiert:

```
<div id="wrapper">
     <div id="head">
          <div id="logo"></div>
     </div>
    <div id="left"></div>
    <div id="content"></div>
```

Das *wrapper*-Div-Element hat inhaltlich keine Bedeutung und ist im Grunde eine Doppelung von body, erweist sich aber beim Layouten mit CSS als sehr praktisch. Dieses Element soll als eine Art „Schutzumschlag“ für alle andere Div-Elemente dienen.

4.1.5 Das HTML-Gerüst

Style-Sheet

Das Styling der einzelnen Elemente geschieht über die css-Datei `template.css`, welche sich in dem *„css“*-Ordner befindet. Sie wird über das <link>-Element eingebunden.

```
<link type="text/css" href="css/template.css"
rel="stylesheet" media="screen" />
```

Als Wert des *media*-Attributes wird *screen* gewählt, da es sich bei dem Webprojekt um eine Bildschirmausgabe handelt.

Die template.css besteht aus einer Vielzahl von Selektoren, denen Eigenschaften zugewiesen werden. Den Eigenschaften werden wiederum Werte zugewiesen.

CSS-Reset

Um die Browservorgaben für die Anzeige der Elemente zu entfernen, wird folgendes *CSS-Reset verwendet.*

```
* {
padding:0;
margin:0;
}
```

Hier werden mittels 2 Eigenschaften die unterschiedlichen padding und margin-Werte der Browser zurückgesetzt. Ein umfangreicheres *CSS-Reset* ist nicht vorgesehen, da dies das Ausmaß dieses Webprojektes nicht erfordert.

Body

Dem Selektor-<Body> werden allgemeine Eigenschaften, wie Hintergrundfarbe, Schriftgröße und Schriftart zugewiesen. Außerdem wird ein Hintergrund, bestehend aus einer 68x68 Pixel großen png-Datei, welche durch den Wert `repeat` endlos wiederholt wird, zugewiesen.

```
body {
      background:url(../images/back.png) repeat;
      font-size: 11px;
      font-family: Verdana, Arial, SunSans-Regular,
      Sans-Serif;
}
```

#wrapper

Der wrapper-div wird durch `margin:0 auto;` zentriert und bekommt eine feste Breite von 960px zugewiesen. Auch hier wird, wie schon bei dem `body`-Element, dem Hintergrund eine 960x68 Pixel große png-Datei zugewiesen. Hier wird das png nur auf der y-Achse wiederholt.

```
#wrapper {
      background:url(../images/wrapper2.png) repeat-y;
      width: 960px;
      margin:0 auto;
}
```

#head

Dem head-div wird eine feste Höhe von 200px zugewiesen. Außerdem wird noch die Eigenschaft `overflow` mit dem Wert `hidden` vergeben, um zu gewährleisten, dass der Inhalt abgeschnitten wird, wenn er die Grenzen des Elements überschreitet.

```
#head {
      height:200px;
      overflow:hidden;
}
```

#logo

Dem logo-div wird eine feste Breite und Höhe zugewiesen. Hinzu kommen noch `margin`-Elemente, welche der Positionierung dienen. Ein Hintergrund bestehend aus einer 132x24 Pixel großen png-Datei wird ebenfalls zugewiesen.

```
#logo{
          margin-top:100px;
          margin-left:207px;
          width:132px;
          height:24px;
          background:url(../images/logo.png) top
no-repeat;
          }
```

#content

Dem content-div wird eine feste Breite, eine Poitionierung mittels `float,` sowie eine Schriftfarbe zugewiesen.

```
#content {
          width:600px;
          float:left;
          color:#FFF;
          }
```

div#content

Die Höhe des content-div ist mit `min-height` definiert, so dass der Inhaltsbereich immer länger als der Navigationsbereich ist. Da der Internet Explorer auf Mac-Systemen keine `min-height` bietet, muss eine Expression zugewiesen werden.

```
div#content {
               min-height:600px;
               height:expression(this.scrollHeight > 600 ?
          "auto":"600px");
               }
```

#left

Dem left-div wird eine feste Breite, eine Poitionierung mittels `float,` sowie eine Schriftfarbe zugewiesen

```
#left {
float: left;
width: 178px;
}
```

#footer

Dem footer-div wird eine feste Breite und Höhe zugewiesen. Außerdem wird mit der Eigenschaft `clear` und dem Wert `both` bewirkt, dass dieses Element weder links noch rechts neben einem vorhergehenden Element stehen kann. Es wird auch mittels `text-align:right;` eine horizontale Textausrichtung für das Element zugeschrieben. Die `padding` Eigenschaften sollen den Text innerhalb des footer-div mittig positionieren. Eine Farbe wurde auch zugewiesen.

```
#footer {
          clear: both;
          height:50px;
          width:685px;
          text-align: right;
          background:#edebe5;
          margin-left:178px;
          padding-top:8px;
          padding-right:10px;
          }
```

4.1.6 Die templateDetails.xml-Datei

Die `templateDetails.xml`, welche für die Installation des Templates wichtig ist, beinhaltet sämtliche Metadaten für unser Webprojekt. Bei der Namensgebung der Datei, wird darauf geachtet, dass das D bei templateDetails, groß geschrieben werden muss. In dem <file> Tag, werden alle für die Installtion benötigten Dateien mit Pfadangabe beschrieben. Die Positionen wo der dynamische Inhalt dargestellt wird wird in dem <positions>-Tag beschrieben.

```
<files>
    <filename>css/template.css</filename>
    <filename>index.php</filename>
    <filename>templateDetails.xml</filename>
</files>
<positions>
    <position>head</position>
    <position>left</position>
    <position>content</position>
    <position>footer</position>
</positions>
```

Es werden auch noch weitere Metadaten, wie Autor, Version oder Erstelldatum mittels XML-Tags beschrieben.

4.1.5 Die Template-Installation

Bevor das Template installiert werden kann, muss die `index.html` in eine `index.php` abgeändert werden. Dies bedeutet nicht, dass die Syntax der `index.html` geändert wird. Joomla! verlangt nach einer index.php, sodass dies notwendig ist, auch bevor man den php-Code in die Datei einträgt.

Nun, da alle Verzeichnisse sowie Dateien, welche für die Installation des Templates benötigt werden, erstellt wurden, gilt es das Template in Joomla! zu installieren.

Die Installation erfolgt im Backend-Bereich von Joomla!. Über Erweiterungen/Templates, gelangt man in den Konfigurationbereich für die sich in dem Template-Verzeichnis befindlichen Webprojekte. Hier erscheint das Template-Praktikum und wird durch Auswahl und auf „Standard“ setzen aktiviert.

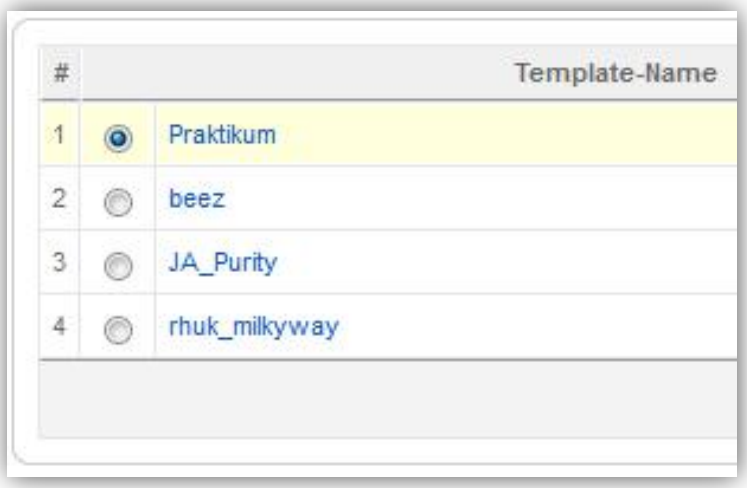


Abbildung 12: Template Auswahl

4.1.7 Erstellen der Webseitennavigation

Die Navigation der Webseite wird aus den vier Menüpunkten „Home, Blog, Pics und Kontakt“ zusammengesetzt. Die Menüpunkte werden durch Joomla! in einer Aufzählungsliste ausgegeben. Diese Liste wird mittels css-Befehlen in der `template.css` gestylt.

#left ul

Die Liste, welche sich innerhalb des left-div befindet, wird mit der Eigenschaft `margin-top` positioniert. Der Menütext wird mit `text-align:right;` rechts ausgerichtet. Außerdem wird mit der Eigenschaft `list-style` und dem Wert `none` die Listendarstellung deaktiviert.

```
#left ul{
          list-style:none;
          text-align:right;
          margin-top:100px;
```

#left li

Die Listenelemente werden mit den `padding` ausgerichtet. Die Größe der einzelnen Listenelemente wird ebenso festgelegt. Die Zeilenhöhe wird mit `line-height` auf 15 Pixel gesetzt und die Schriftgröße auf 16 Pixel.

```
#left li{
     padding:13px  5px  0px  0px;
     height:26px;
     line-height:15px;
     font-size:16px;
```

#left li a

Mit der Eigenschaft `color` wird hier der Menütext der nicht aktiven Menüpunkte farbig gestaltet. Durch die Eigenschaft `text-decoration` und dem Wert `none` wird der Unterstrich unter dem Text deaktiviert.

```
#left li a{
          color:#edebe5;
          text-decoration:none;
          }
```

#left li#current

Hier werden die aktiven Menüelemente mittels `background-color` mit einem farbigen Hintergrund gestaltet.

```
#left li#current{
               background-color:#edebe5
               }
```

#left li#current a

Hier wird der Text der aktiven Menüpunkte anders farbig gestaltet.

```
#left li#current a{
                   color:#d4643b;
                   }
```

#left li#current a:hover

Bewegt der Nutzer die Maus über die nicht aktiven Menüpunkte, wird dies mittels eines `hover`-Effekts visualisiert. Der hover-Effekt stellt eine farbliche Veränderung des Menütextes dar.

```
#left li a:hover{
          color:#d4643b;
          }
```

Bevor die Menüpunkte in dem Backend-Bereich in Joomla! erstellt werden können, müssen noch Beiträge erstellt werden, da den Menüpunkten ein Beitrag zugeordnet werden muss. Dafür werden über das Menü *Inhalt->Beiträge* neue Beiträge erstellt. Hierfür werden „nicht kategorieste“ Beiträge abgefasst. Dies bedeutet, dass die Beiträge keinem Bereich und somit auch keiner Kategorie zugeordnet sind. Der Inhalt des Textes dieser Beiträge spielt dabei auch keine Rolle. Es muss lediglich Text in den Beiträgen vorhanden sein. Bis auf den Beitrag der Startseite, wird kein anderer Beitrag für die Startseite freigegeben werden.

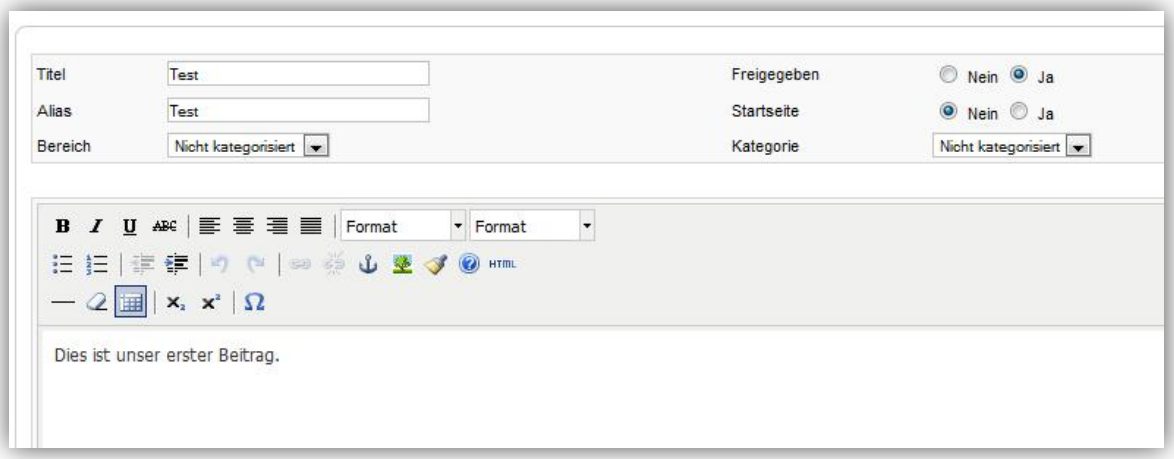


Abbildung 13: Beitragerstellung

Nun werden die Menüpunkte in der Konfiguration des Hauptmenüs, in welches man über *Menüs->Hauptmenüs* gelangt, angelegt. Für die Menüpunkte wird der Menütyp „Layout: Beitrag (Joomla! Standard) gewählt.

Abbildung 14: Menütyp

In den Menüeintrags-Einstellungen wird für die einzelnen Menüeinträge ein Titel sowie ein Alias vergeben.

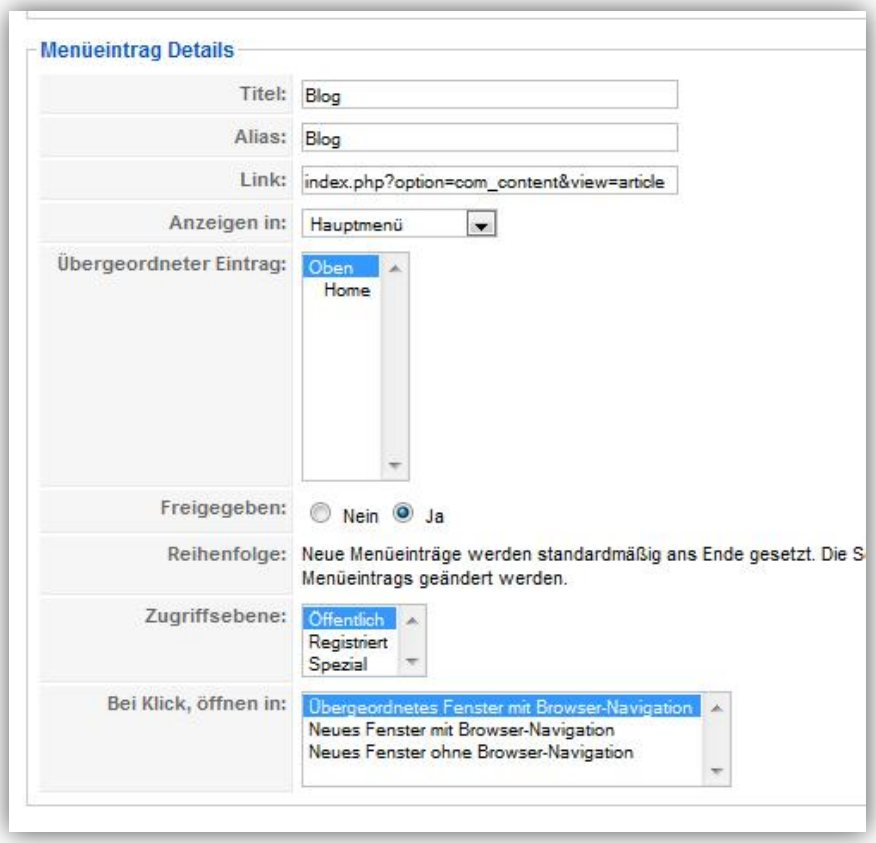


Abbildung 15: Menüeinstellungen

Um das Hauptmenü zu aktivieren, bedarf es noch 3 weiterer Schritte. Der erste Schritt ist die Aktivierung des Moduls „Hauptmenü“ in dem Menü *Erweiterungen->Module.* Hier wird das Modul aktiviert. In den Einstellungen dieses Moduls muss

noch die Position *„left“* gewählt werden, damit das Hauptmenü auch in dem gewünschten Bereich dargestellt wird.

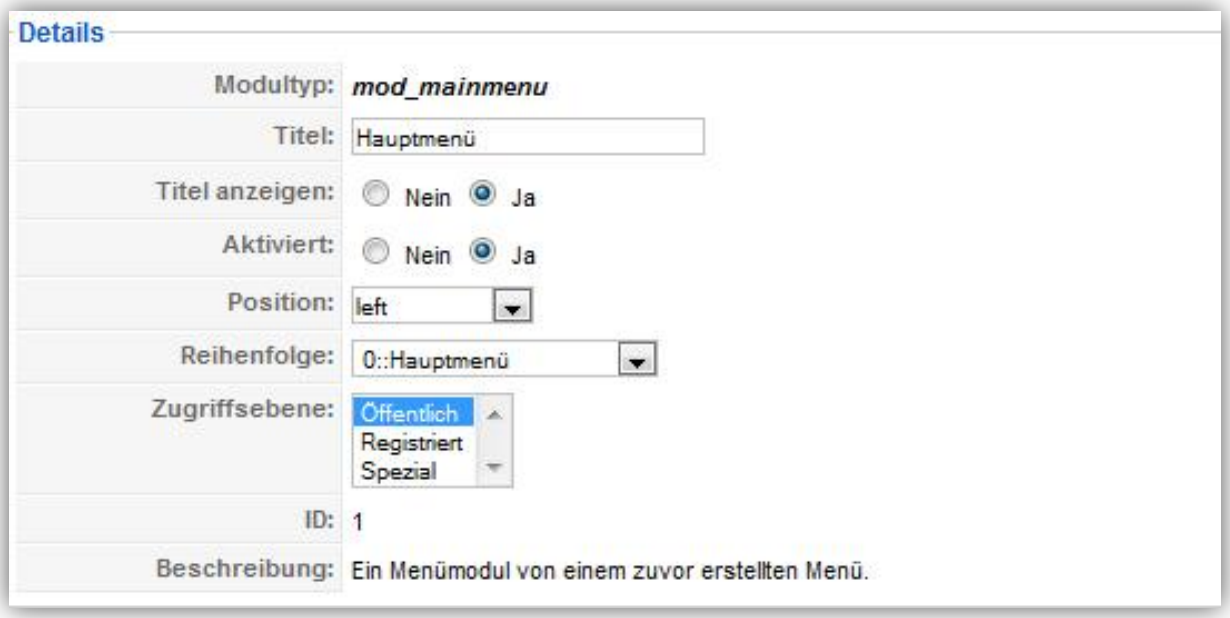


Abbildung 16: Modul_Mainmenü einstellungen

Nun, da alle Einstellungen getroffen sind und die Menüpunkte angelegt sind, muss ein jdoc-Befehl in die `index.php` integriert werden. Dieser jdoc-Befehl wird innerhalb des `left-div` angelegt.

```
<div id="left"><jdoc:include type="modules" name="left"
style="none" /></div>
```

Bei der Vorschau des Templates erscheint nun die Navigation.

Abbildung 17: Navigation

4.1.8 Dynamische Inhalte erstellen

Metadaten

Um Metadaten für die Webseite auszugeben, muss in der `index.php` innerhalb des <head> - Tags folgende jdoc-Anweisung eintragen werden.

```
<jdoc:include type="head" />
```

Die Ausgabe der durch Joomla! generierten Metadaten sieht wie folgt aus:

```
<meta http-equiv="content-type" content="text/html;
charset=utf-8" />
<meta name="robots" content="index, follow" />
<meta name="keywords" content="joomla, Joomla" />
<meta name="description" content="Joomla! - dynamische
Portal-Engine und Content-Management-System" />
<meta name="generator" content="Joomla! 1.5 - Open Source
Content Management" />
```

Slideshow

Bei der Slideshow kommt ein externes Modul namens „*Nivo-Slider“* zum Einsatz. Es handelt sich hierbei um eine Slideshow, welche manuell bedient werden kann oder aber die Bilder auch automatisch in bestimmten Abständen ändern kann. Für das Erstellen der Slideshow muss zunächst ein „Modul“ installiert werden.

Die Modulinstallation von Joomla! befindet sich in dem Backend-Bereich, im Menü *Erweiterungen->Installieren/Deinstallieren*.

In dem Bereich „*Paketdatei hochladen*“ wählt man zunächst das „*mod_nivoslider.rar*“ aus und bedient die Schaltfläche „*Datei hochladen & installieren“.*

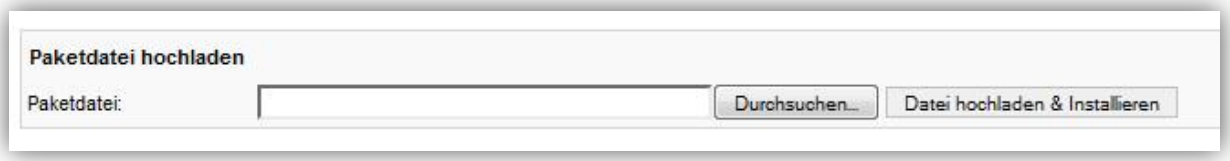


Abbildung 18: Paketdatein installieren

Nach erfolgreicher Installation befindet sich das Modul in *Erweiterungen->Module*.

Nun wird das Modul aktiviert und dessen Einstellungen bearbeitet. Die Titel-Anzeige wird deaktiviert und es muss wieder eine Position für das Modul bestimmt werden. Die Position für das Modul Nivo-Slider wird auf *content* gestellt.

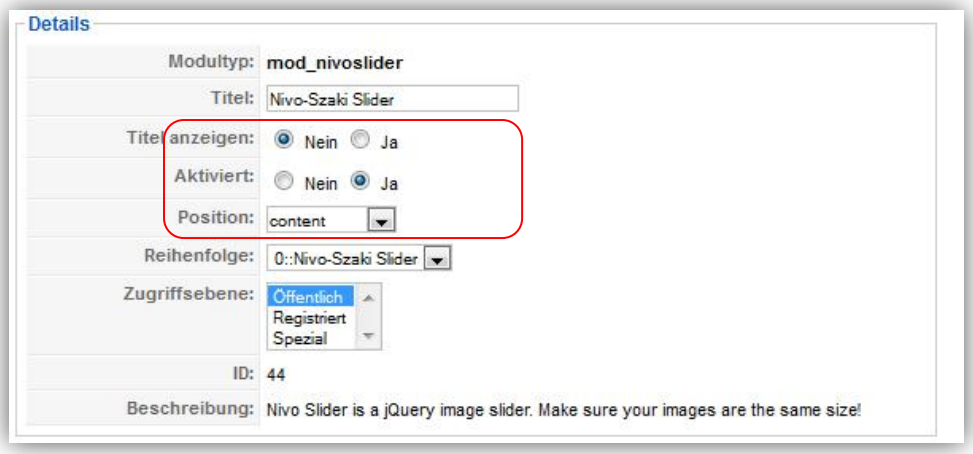


Abbildung 20: slider

Die Menüzuweisung wird auf „Home“ gestellt, da die Slideshow nur auf der Startseite angezeigt werden soll.

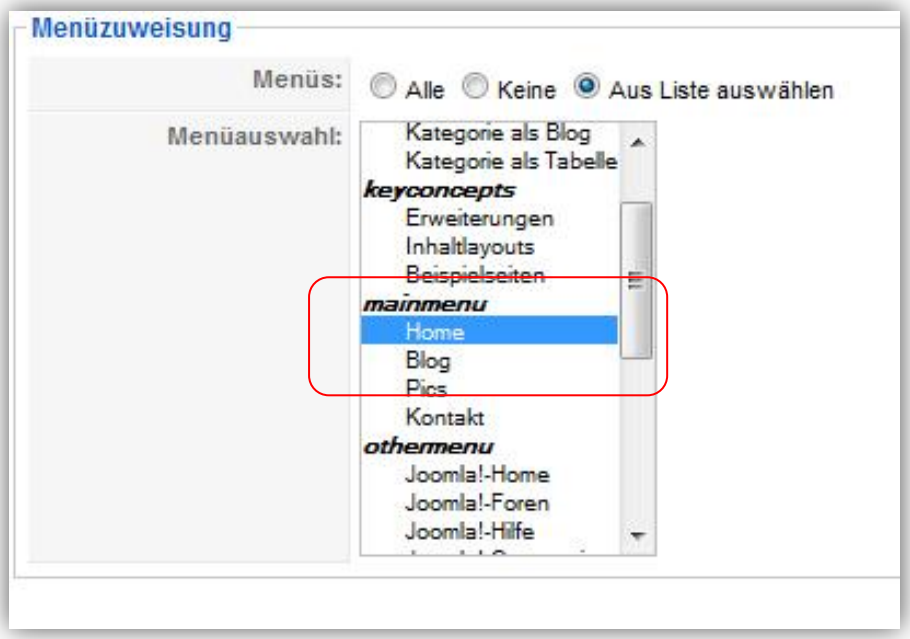


Abbildung 21: Menüzuweisung

Die Bilder für die Slideshow müssen in dem *Medien-Bereich* von Joomla! im Ordner „banners“ hochgeladen werden. Für die endgültige Implementierung der Slideshow, wird ein jdoc-Befehl innerhalb des `content-div` eingefügt.

```
<jdoc:include type="component" style="xhtml" />
```

Beitrag auf der Startseite

Unterhalb der Slideshow auf der Startseite, soll ein Beitrag erscheinen. Um Beiträge auf der Webseite anzeigen zu lassen,ist eine weitere jdoc-Anweisung nötig. Diese wird innerhalb des <content>-Tag eingetragen.

<jdoc:include type="modules" name="content" style="xhtml" />

```
<jdoc:include type="modules" name="content" style="xhtml" />
```

Jetzt wird auf der Startseite unterhalb der Slideshow der Seitentitel “Willkommen auf der Startseite“ dargestellt. Dieser wird von dem Menüeintrag „Home“ (Startseite) erzeugt und muss in den Systemparameter „Einstellungen“ des Menüeintrags deaktiviert werden.

Abbildung 22: Systemparameter

Für den Begrüßungstext, der unterhalb der Slideshow erzeugt werden soll, wird ein neuer Beitrag benötigt. Dieser soll „nicht kategorisiert“ sein. Er wird für die Startseite freigegeben sein. Damit erscheint er nur auf der Startseite. Für den Beitrag wird ein Titel vergeben, welcher als Überschrift dienen soll. Zudem wid in das Textfeld ein beliebiger Text eingetragen. Hierbei empfiehlt sich ein *Dummytext-Generator.* In den erweiterten Parametern wird das pdf-Icon, Drucken-Icon und Email-Icon deaktiviert. Die Ausgabe für den „Autor“, „Erstellungsdatum und –zeit“ und „Bearbeitungsdatum und –zeit“ wird ebenfalls in den erweiterten Parametern deaktiviert.

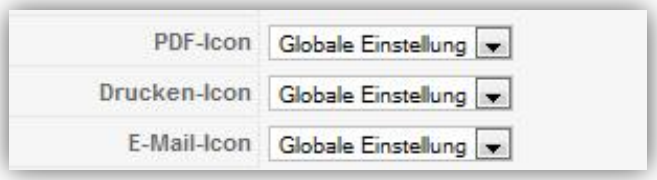


Abbildung 22: weitere Parameter

Auf der Startseite erscheint nun unterhalb der Slideshow der eben erstellte Beitrag. Allerdings noch ohne jegliche Formatierung mittels CSS, da den Textelementen bisher noch keine Style-Angaben zugewiesen wurde.

Nun wird der Inhaltstext mittels CSS mit Style-Elementen versehen. Für die Klassenvergabe werden die Klassen verwendet, welche Joomla! ausgibt. Diese werden mittels Quellcode-Anzeige im Browser ersichtlich.

Um die Beitragsüberschrift per CSS zu bearbeiten, wird folgender Quellcode in die „*template.css“* eingefügt:

```
.contentheading{
                font-size:24px;
                border-bottom:solid;
                border-color:#F60;
```

Abschließend wird nun die Position des Textes justiert.

```
.contentpaneopen{
                margin-left:30px;
                  }
```

Bloginhalt für den Menüpunkt Blog

Die Erstellung von Beiträgen für Bloginhalte verlangt nach einem „Bereich“ und einer „Kategorie“. Die Bereichserstellung erfolgt über *Inhalt->Bereiche und* die Erstellung von Kategorien über *Inhalt->Kategorie.* Nun wird der Beitrag z.B. mit dem Titel „Mein erster Blogbeitrag“ erstellt und einem Bereich sowie einer Kategorie zugeordnet.

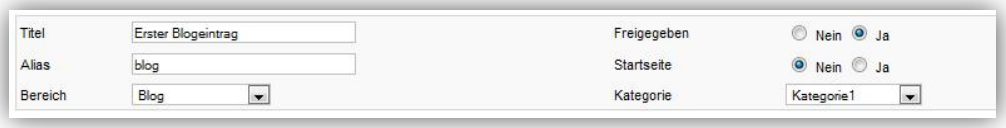


Abbildung 23: Titelvergabe

Wichtig ist hier wieder, darauf zu achten, dass der Beitrag nicht für die Startseite freigegeben ist. Der Inhaltstext wird abermals von einem Dummytext-Generator erstellt und in das Textfeld eingefügt.

Um eine „Weiterlesen“-Funktion zu implementieren, muss der Textcursor an der gewünschten Stelle im Text platziert werden und danach auf den „Weiterlesen“-Button geklickt werden. Dadurch erscheint an der Stelle, wo sich der Textcursor

befand, eine rot-gestrichelte Linie. Diese zeigt an, bis zu welcher Stelle der Text in der ersten Ansicht gehen wird.

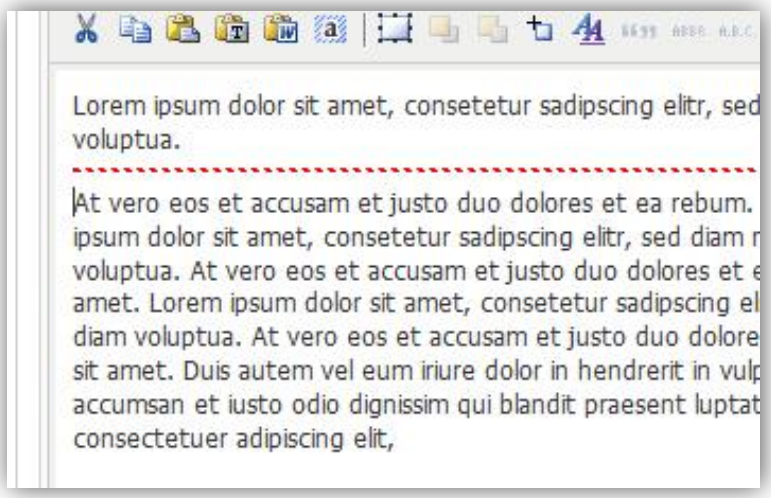
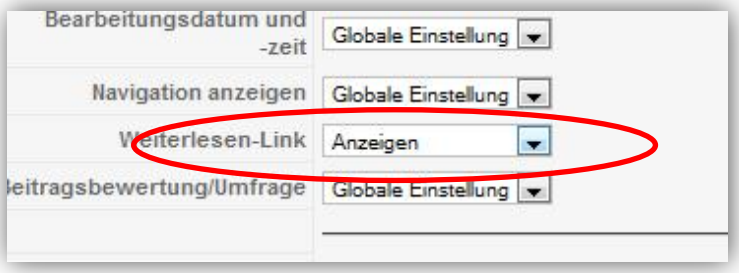


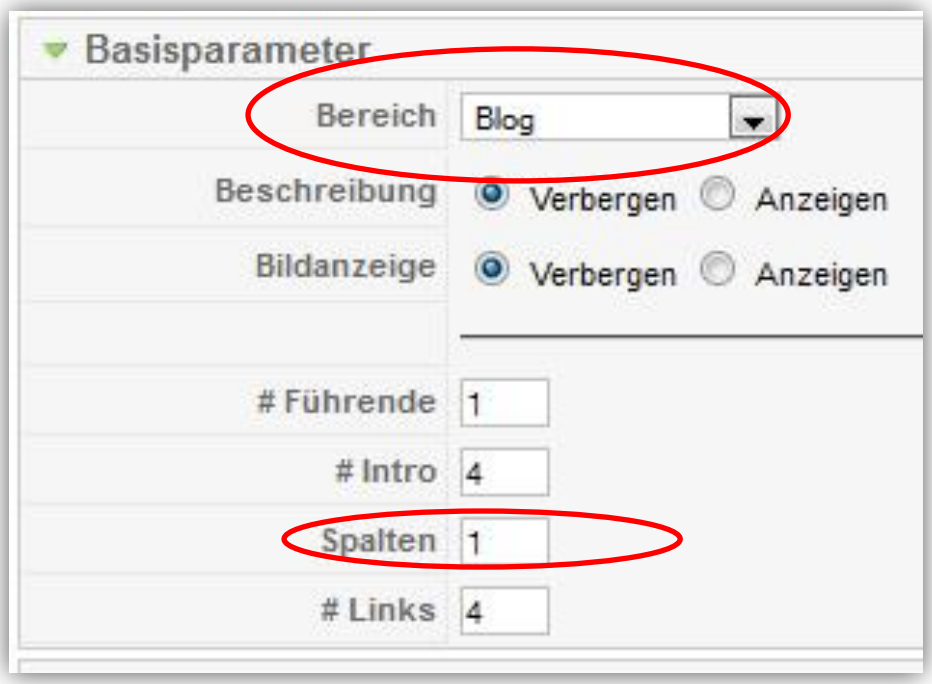
Abbildung 23: weiterlesen

Um die „Weiterlesen“-Funktion zu aktivieren, muss der Beitrag *Menütyp* von dem Menüeintrag „Blog“ in ein „Layout: Bereichs-Blog“ geändert werden.

Nun befindet sich auf der rechten Seite, in den Menüeintrags-Einstellungen, in der Liste der Komponentenparameter die Einstellung für den *Weiterlesen-Link.* Dieser wird nun aktiviert.



In der Liste der Komponentenparameter befindet sich ebenso der Parameter „Bearbeitungsdatum und –zeit“, der deaktiviert wird. In den Basisparameter-Einstellungen muss der Bereich ausgewählt werden, dem auch der Beitrag, der zur Anzeige gebracht werden soll, zugeordnet ist. Um ein einspaltiges Bloglayout zu erstellen, muss der Wert bei Spalten auf „1“ gesetzt werden. In den Systemparamter-Einstellungen wird nun noch der Seitentitel deaktiviert.



Des Weiteren müssen die Komponentenparameter „Autor“ sowie „Erstelldatum und –zeit“ mittels `css` angepasst werden. Für diese zwei Parameter vergibt Joomla! Klassen, welche in die `template.css` eingefügt werden.

```
.createdate{
          font-size:10px;
          font-style:italic;
          }
.small{
          font-size:10px;
          font-style:italic;
     }

.readon{
     color: #69F;
     }
```

Bildergalerie für den Menüpunkt „pics“

Für die Bildergalerie wird ein exterens Plugin namens „SIGE“ (Simple Image Gallery Extended) verwendet. Dieses wird über *Erweiterungen->Installieren/Deinstallieren* in dem Installationsbereich installiert.

Danach wird das Plugin unter *Erweiterungen->Plugins* installiert. Nun werden in dem *Medienbereich* von Joomla! im Ordner „stories“ zwei Unterordner mit der Bezeichnung „Album1“ und „Album2“ erstellt und diese mit Bildern für die Gallerie gefüllt. Nun wird in dem Beitrag „pics“ Folgendes in das Textfeld geschrieben:

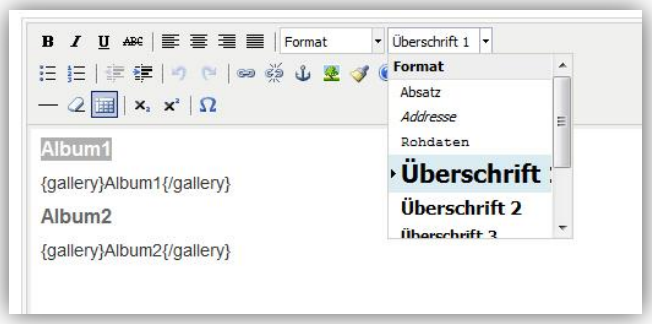
```
Album1

{gallery}Album1{/gallery}

Album2

{gallery}Album2{/gallery}
```

Für die Galeriebezeichnungen *Album1* und *Album2* wird in dem Texteditor unter Format der Punkt „Überschrift 1“ ausgewählt.



Nun wird noch der *Seitentitel* in der Beitragseinstellung des Beitrags „pics“ deaktiviert.

Abbildung 29: Gallerie

Die Gallerie zeigt die Bilder als Thumbnailvorschau. Durch klicken auf ein Bild, öffnet sich ähnlich wie bei dem „lightbox“ oder „fancybox“- System das Bild in einem seperaten Fenster.

Kontaktformular für den Menüpunkt Kontakt

Um das Kontaktformular zu erstellen, werden zuvor die Einstellungen des Standart-Kontakt „Name“ unter *Komponenten->Kontakte->Kontakte,* bearbeitet.

Hier werden alle erforderlichen Daten zur Person eingetragen. Dabei gibt es allerdings keine Pflichtfelder. Falls bestimmte Felder nicht mit Informationen gefüllt werden, sollten diese noch in den *Kontaktparametern* deaktiviert werden. Außerdem wird der Menütyp unter *„Menüs->Hauptmenü->Kontakt->Typ bearbeiten“ in den Typ* „Layout: Kontakt (Joomla!-Standard)“ geändert.

Klicken sie in dem Feld Menütyp auf den Butten *„Typ bearbeiten*“. Wählen sie hier nun unter *„Kontakte->Kontakt“* den Typ „Layout: Kontakt (Joomla!-Standard)“ aus.

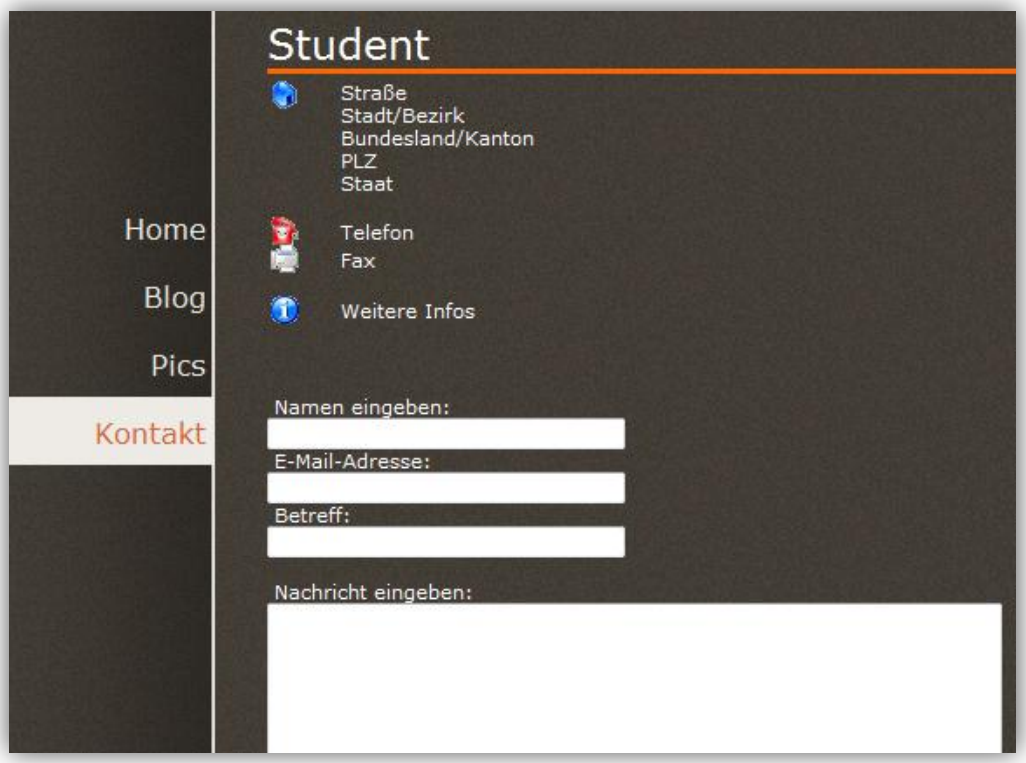


Abbildung 30: Kontakt

Footer Implementierung

Nun, da alle Menüpunkte mit dynamischen Inhalten gefüllt sind, wird der Footer unter Zuhilfename_eines Moduls erzeugt. Das Modul trägt die Bezeichnung „Fußzeile“ und muss in „Erweiterungen->Module“ aktiviert werden.

Um die Ausgabe des Footerinhalts zu erzeugen, wird ein jdoc-Befehl in die `index.php` eingefügt. Der Befehl wird innerhalb des `footer-div` geschrieben.

```
<jdoc:include   type="modules"   name="footer"
```

Abbildung 31: Footer

4.2 Erzeugung der Praktika

Das Template für Joomla! wurde in Punkt 4.1 erfolgreich erzeugt. Nun, geht es darum die 3 Praktika zu erzeugen. Wie schon in der Planung besprochen, soll das Komplexpraktikum in 3 Teile unterteilt werden.

In dem ersten Praktikumsteil wird den Studenten, welche das Praktikum durchführen sollen, mittels einer Einleitung beschrieben, welche Ziele das Komplexpraktikum hat. Damit soll erreicht werden, dass der Student sich ein Bild davon machen kann, was Ihn in den Praktika erwartet. Dabei wird zu Beginn des Praktikums, mittels Grafiken das gewünschte Endergebnis aufgezeigt. Außerdem wird dem Studenten die Visualisierung der Arbeitsschritte des Praktikums beschrieben.

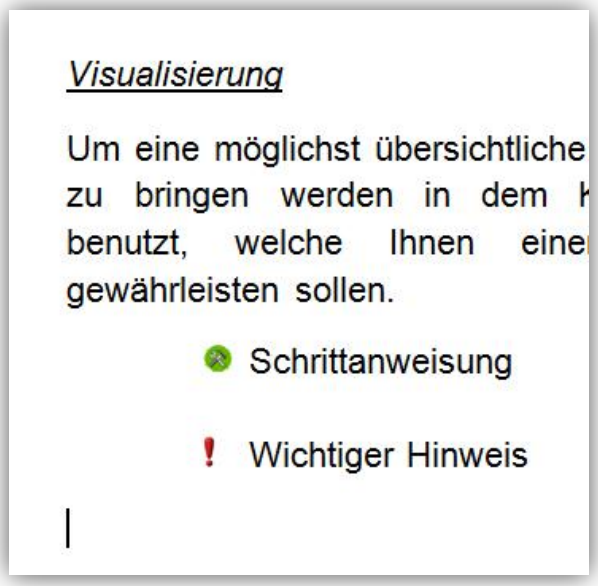


Abbildung 32: Protokol

Damit soll den Studenten ersichtlich werden, welche Symbole für welchen Inhalt stehen.

Die Planung sieht vor die lokale Serverinstallation von XAMPP in dem Praktikum zu beschreiben. Dies wird in einer Installationsanleitung, welche 4-5 Schritte beinhaltet, geschehen. Für diese Anweisungen, werden Grafiken zu Hilfe genommen, welche den Installationsprozess von dem XAMPP darstellt.

Danach wird der Student mit Hilfe vereinzelter Codebeispiele das HTML/CSS-layout erstellen.

In dem 2. Teil des Praktikums, müssen die Studenten das Backend anpassen. Hier werden die einzelnen Schritte für das Löschen von Beiträgen oder Bereichen beschrieben. Außerdem geht es in diesem Teil um die Erstellung der templateDetails.xml. Der Quellcode hierfür liegt dem Studenten vor.

Danach wird mittels verschiedener Grafiken beschrieben, wie das Template in das CMS Joomla! installiert wird. Nach der Installation wird der Student, zum ersten mal mit dynamischen Inhalten arbeiten. Hier muss erwähnt werden, das jdoc-Anweisungen nicht weiter erläutert werden.

Im Verlaufe des Praktikums soll dem Studenten immer weniger Hilfestellung geboten werden.

Abbildungsverzeichnis

Abbildung 1: Aufbau CMS 9
Abbildung 2: CSS layout 13
Abbildung 3: wrapper-breite 15
Abbildung 4: screendesign 16
Abbildung 5: Datenbank 27
Abbildung 6: joomla Installation 28
Abbildung 7: Backend Anmeldung 29
Abbildung 8: Module 30
Abbildung 9: Butten für aktivieren / Deaktivieren 30
Abbildung 10: Seo_URLS 31
Abbildung 11: Verzeichnisstruktur 32
Abbildung 12: Template Auswahl 41
Abbildung 13: Beitragerstellung 44
Abbildung 14: Menütyp 45
Abbildung 15: Menüeinstellungen 45
Abbildung 16: Modul_Mainmenü einstellungen 46
Abbildung 17: Navigation 46
Abbildung 18: Paketdatein installieren 48
Abbildung 20: slider 48
Abbildung 21: Menüzuweisung 49
Abbildung 22: Systemparameter 50
Abbildung 22: weitere Parameter 50
Abbildung 23: Titelvergabe 51
Abbildung 23: weiterlesen 52
Abbildung 29: Gallerie 56
Abbildung 30: Kontakt 57
Abbildung 31: Footer 58
Abbildung 32: Protokol 59

Anhang A

Index.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head><jdoc:include type="head" />

<link type="text/css" href="/Joomla/templates/Praktikum/css/template.css" rel="stylesheet" media="screen" />

<title></title>

</head>

<body>

<div id="wrapper">

    <div id="head"><div id="logo"></div></div>

    <div id="left"><jdoc:include type="modules" name="left" style="none" /></div>

    <div id="content"><jdoc:include type="modules" name="content" style="xhtml" />

    <jdoc:include type="component" style="xhtml" />

      </div>

 <div id="footer"><jdoc:include type="modules" name="footer" style="none" /></div>

</div>

</body>

</html>
```

Anhang B

CD

Literarturverzeichnis

1. Internetquellen

http://openbook.galileocomputing.de/joomla15/ (Zugriff 16.08.2011)

http://extensions.joomla.org/ (Zugriff 17.10.2011)

2. Printmedien

Addison Wesley(2005), Hagen Graf, *Joomla 1.5*

Niegemann, H. M. (2008). *Kompendium multimediales Lernen*. Berlin Heidelberg: Springer-Verlag.

Schott, F. (1991). Instruktionsdesign, Instruktionstheorie und Wissensdesign: Aufgabenstellung, gegenwärtiger Stand und zukünftige Herausforderungen. *Unterrichtswissenschaft*, 19 (3), 195-217.

Klauer, K. J. & Leutner, D. (2007). *Lehren und Lernen: Einführung in die Instruktionspsychologie.* Basel: Beltz Verlag.

Weidenmann, B. (1993). Informierende Bilder. In B. Weidenmann (Hrsg.). *Wissenserwerb mit Bildern: instruktionale Bilder in Printmedien, Film/Video und Computerprogrammen.* S. 9-58. Bern: Huber.

Eidesstattliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.

Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Chemnitz, den 11. November 2011

Toni Niemeier