Schriftliche Hausarbeit
zur Abschlussprüfung
der erweiternden Studien für Lehrer
im Fach Informatik

VII. Weiterbildungskurs
in Zusammenarbeit mit der
Fernuniversität Hagen

Eingereicht dem Amt für Lehrerausbildung -Außenstelle Gießen -
Vorsitzender des Prüfungsausschusses: Jungermann

Contentmanagement mit Server Side Includes, PHP und MySQL

Material für Projektarbeit in der Oberstufe

Verfasser: Hermann-Josef Wehner

Gutachter: StD Otto Wehrheim

Inhaltsverzeichnis

1. Das WWW 1993 und 2003

Der Umfang der Informationen, die im Worldwide Web angeboten werden, hat sich in den letzten Jahren vervielfacht und ein Ende der Entwicklung ist derzeit nicht abzusehen. Die Zahl der Einträge in Suchmaschinen belegt dies: Im Juli 2003, ca. zehn Jahre nach der Entstehung des WWW, gab es mehr als 42 Millionen Websites[1]. Im Juni 1993 bestand das WWW lediglich aus 130 Sites[2]. Von einem Medium zur Verbreitung von Forschungsergebnissen und wissenschaftlicher Literatur hat sich das World Wide Web zu einem Massenmedium entwickelt, das für viele aus dem täglichen Leben nicht mehr wegzudenken ist. Zur interessanten Geschichte des WWW finden sich detaillierte Informationen auf den Seiten des CERN[3], wo Tim Berners-Lee das WWW um 1990 entwickelte, und bei dem W3-Konsortium[4].

Für das Erstellen von Web-Dokumenten gab es bereits 1992 einen HTML-Editor von Tim Berners-Lee, der der Bedienungsanleitung zufolge recht komfortabel war[5]. Allerdings war das Angebot an Websites noch so überschaubar[6], dass verschiedenste Methoden vorgeschlagen wurden, um das in der Entstehung begriffene WWW mit Inhalten zu bereichern[7].

Auch heute kann man ein HTML-Dokument noch immer mit einem einfachen Texteditor erzeugen. Allerdings kommt diese Methode allenfalls für Inhalte in Frage, die sich über einen längeren Zeitraum nicht ändern. Wer aktuelle Informationen anbieten will, kann nicht bei jeder Änderung mehrere Dokumente manuell anpassen. Insbesondere Unternehmen, deren Existenz maßgeblich von der Internetpräsenz abhängig ist, wie beispielsweise Online-Warenhäuser, Nachrichtenportale oder Auktionshäuser, müssen für die Erstellung und Wartung ihrer Websites andere Methoden anwenden.

1.1 Manuelle Seitenerstellung und automatisiertes Contentmanagement

Tim Berners-Lee geht in seinen oben erwähnten Ratschlägen noch von der Personalunion von Serverbetreiber, Autor und Webmaster aus, denn er konnte kaum die Entwicklung voraussehen, die das WWW in den letzten Jahren genommen hat. Umfangreiche Webpräsenzen unserer Tage werden allerdings in der Regel nicht mehr von einer Person erstellt oder gewartet. Allein die Menge an Informationen erfordert Arbeitsteilung.

Dadurch ergeben sich für die Praxis enorme Schwierigkeiten:

- Verweise innerhalb der Seiten müssen ständig aktualisiert werden. Wird eine Seite entfernt oder an eine andere Stelle verlagert, müssen alle Links auf diese Seite angepasst werden. Entsprechendes gilt, wenn neue Seiten publiziert werden.
- Die Kontrolle über die Inhalte liegt bei jedem einzelnen Autor. Die Wahrscheinlichkeit, dass Beiträge ungewollt veröffentlicht werden, steigt mit dem Zeitdruck, unter dem publiziert werden muss.
- Jeder Autor muss über HTML-Kenntnisse verfügen, um seine Seiten erstellen zu können. Ist dies nicht der Fall, müssen Inhalte einem HTML-kundigen Webmaster übergeben werden, der sie in

1 http://news.netcraft.com/archives/web_server_survey.html
2 http://www.mit.edu/people/mkgray/net/web-growth-summary.html
3 http://public.web.cern.ch/public/about/achievements/www/history/history.html
4 http://www.w3.org/History
5 http://www.w3.org/History/1991-WWW-NeXT
http://www.w3.org/MarkUp/tims_editor
6 http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html
http://www.w3.org/History/19921103-hypertext/hypertext/DataSources/WWW/Servers.html
7 http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Provider/Overview.html

web-gerechte Form überführt. Dabei können leicht ungewollte inhaltliche Veränderungen entstehen.

- Bei umfangreichen Veröffentlichungen verliert man leicht die Übersicht. Nicht mehr aktuelle Seiten können übersehen werden.
- Durch die Verwendung eines „Corporate Design“ entsteht zusätzlicher Arbeitsaufwand. Schriftgröße, Schriftart, Layout etc. müssen vereinbarungsgemäß überall in gleicher Weise angewendet werden. Anpassungen des Designs erfordern umfangreiche Änderungen in einzelnen Dateien.

Um diese Schwierigkeiten zu umschiffen, werden für die Gestaltung umfangreicher Websites mit vorwiegend dynamischem Inhalt seit mehreren Jahren so genannte Content-Management-Systeme (CMS) verwendet.

1.2 Eigenschaften von Content-Management-Systemen

Eine herausragende Eigenschaft von Content-Management-Systemen[8] ist die Trennung von HTML-Programmierung und Inhalt. Autoren müssen Beiträge lediglich in einem Formular eintragen. Das endgültige Erscheinungsbild der Seite wird von vorgefertigten Schablonen (Templates) bestimmt. Moderne CMS bieten sogar WYSIWYG-Editoren an, wodurch man beim Erstellen neuer Webseiten auf den Komfort einer Textverarbeitung zurückgreifen kann.

Professionelle Systeme zeichnen sich in der Regel durch folgende Eigenschaften aus:

- Beim Erstellen von HTML-Seiten wird Zeit gespart, weil der Autor sich völlig auf den Inhalt konzentrieren kann. HTML-Kenntnisse sind nicht mehr erforderlich.
- Inhalte können in verschiedenen Formaten (Word, XML, HTML, TXT) importiert werden, damit Autoren in ihrer gewohnten Umgebung bleiben können.
- Die Kontrolle der Inhalte kann durch Chefredakteure übernommen werden, die einen Artikel zur Veröffentlichung freigeben oder bei Bedarf an den Autor zur Überarbeitung zurückschicken können.
- Eine Benutzerverwaltung gewährleistet, dass jeder Mitarbeiter nur die Bereiche ändern kann, für die er autorisiert ist. Versehentliche oder absichtliche Sabotage wird dadurch erschwert. Autoren können so leicht ihre eigenen Beiträge aktualisieren, nicht aber die anderer.
- Veröffentlichungstermine und Zeiträume können individuell festgelegt werden. Damit ist gewährleistet, dass nur aktuelle Informationen präsentiert werden.
- Ein Versionshistorie erlaubt, den Entstehungsprozess von Beiträgen zu verfolgen und gegebenenfalls auf alte Versionen zurückzugreifen.
- Hyperlinks innerhalb der Dokumente werden automatisch angepasst, wenn Beiträge gelöscht oder innerhalb der Dokumentenstruktur verschoben werden.
- Navigationselemente werden automatisch generiert. Bei der Änderung des Angebots müssen Inhaltsverzeichnisse nicht manuell geändert werden.
- Komfortable Suchfunktionen lassen Informationen auch in umfangreichen Webpräsenzen leicht finden.
- Automatisch erzeugte Übersichtsseiten (Sitemap) erleichtern die Orientierung.

Durch die genannten Eigenschaften erreicht man in der Regel eine Kostensenkung.

8 Der Begriff wird im Folgenden als CMS abgekürzt.

Die derzeit angebotenen CMS bieten jeweils die genannten Eigenschaften und Funktionen in unterschiedlichem Umfang an. Allen Systemen ist allerdings gemein, dass sie das Veröffentlichen von Inhalten in Netzen vereinfachen. Der Einfachheit halber soll im Folgenden auch von CMS gesprochen werden, wenn zumindest diese Eigenschaft vorhanden ist.

2. CMS im Lehrplan für Informatik

Der hessische Lehrplan für das Fach Informatik sieht in der Jahrgangsstufe 11 das Thema Internet vor[9]. Die Schüler sollen unter anderem ein Informationssystem auf Hypertextbasis erstellen. Ausdrücklich wird darauf hingewiesen, dass Frames und Formulare dabei verwendet werden sollen. Da Formulare ohne Programme, die diese auswerten, sinnlos sind, ist davon auszugehen, dass das System zumindest kurze Programme enthalten soll.

Ein einfaches CMS in dem oben beschriebenen Sinne lässt sich bereits realisieren, wenn man anstelle von individuellen Seiten ein Frameset verwendet und die Inhalte jeweils in einem Frame darstellt, der keine grafischen Besonderheiten enthält.

Komplexe Systeme arbeiten mit Datenbanken, in denen neben den Inhalten das Layout sowie eventuell nötige Programme gespeichert sind. Da man eine Datenbank nicht aus dem reinen HTML-Code heraus abfragen kann, werden Skriptsprachen verwendet, die entweder in den HTML-Code eingebettet werden (z.B. PHP) oder diesen selbst erst erzeugen (z.B. Perl). Zwischen den Extremen sind Abstufungen denkbar: So existieren beispielsweise CMS, die mit Skriptsprachen arbeiten, die Inhalte allerdings als Dateien auf dem Server ablegen[10].

Im weiteren Verlauf der Ausführungen werden wir ein einfaches CMS entwickeln. Ausgehend von einem einfachen Frameset wird durch schrittweises Verändern oder Hinzufügen von Funktionen ein einfaches datenbankgestütztes CMS entwickelt.

Die einzelnen Entwicklungsstufen können an verschiedenen Stellen in Form von Projektarbeit realisiert werden.

2.1 Anpassung des Schwierigkeitsgrades durch Wahl der Programmiermethode

Das von den Schülern in der Stufe 11 erstellte Hypertext-System kann in den folgenden Kurshalbjahren im Sinne eines Spiralcurriculums zu einem mehr oder weniger komfortablen CMS ausgebaut werden. Wann eine komplexere Ausbaustufe in Angriff genommen wird, hängt von der individuellen Unterrichtsgestaltung ab; ein Vorschlag folgt weiter unten.

Folgende Entwicklungsstufen sind denkbar:

1. Frameset (vgl. oben)
2. Frameset mit Submenu
3. Frameset mit Server Side Includes (Direktive #exec)
4. Frameset mit Server Side Includes (Direktive #include)
5. CMS auf Dateibasis unter Verwendung einer Skriptsprache
6. wie 5., zusätzlich mit Benutzerverwaltung und/oder Editorfunktion

9 Lehrplan Informatik des Landes Hessen S. 8
10 Einen Überblick über die verfügbaren Systeme bietet http://www.contentmanager.de/itguide/marktuebersicht.html

7. CMS auf Datenbankbasis unter Verwendung einer Skriptsprache mit Benutzerverwaltung und Redaktionsfunktion

In den folgenden Ausführungen wird jede dieser Ausbaustufen beschrieben. Dabei werden in erster Linie die nötigen Voraussetzungen dargelegt, die zur Programmierung nötig sind. Auf detaillierte Hinweise zur Unterrichtsgestaltung soll verzichtet werden, da dies den Rahmen sprengen würde. Die Formulierung von Aufgabenstellungen hängt sehr von dem vorangegangenen Unterricht ab und muss daher dem Lehrenden überlassen werden.

2.2 Skalierung durch Modularisierung des Systems

Die Erstellung der Inhalte ist von ihrer Darstellung im Browser völlig unabhängig. Es wäre demnach denkbar, dass man ein System erstellt, das lediglich Verzeichnisinhalte automatisch in eine Website integriert. Das Erstellen der Inhalte könnte über ein HTML-Formular geschehen oder durch den Transfer fertiger Dateien mit ftp. Genauso wäre ein System vorstellbar, das seine Inhalte zwar per Skriptsprache aus einer Datenbank liest, bei dem aber die Eingabe der Inhalte nicht über ein eigens programmiertes Autorenmodul stattfindet, sondern manuell. Die Datenbank wird dabei über ein existierendes Frontend (z.B. phpMyAdmin für MySQL) aktualisiert. Ein eigenes Eingabemodul könnte in einer späteren Projektphase hinzugefügt werden.

Auf diese Weise kann ein Projekt auf die Leistungsfähigkeit der Arbeitsgruppe und den zur Verfügung stehenden Zeitrahmen abgestimmt werden.

2.3 Verteilung auf die Kurshalbjahre

2.3.1 Kurshalbjahr 11.1

Im ersten Halbjahr der Stufe 11 können lediglich die ersten oben genannten Ausbaustufen (CMS mit Frames und Submenu oder SSI) realisiert werden. Zwar wird im Lehrplan als fakultativer Unterrichtsinhalt Webdatenbanken und Zugriff mittels Skriptsprachen vorgeschlagen, allerdings erscheint es wenig sinnvoll, diese Kenntnisse bereits für ein CMS-Projekt einzusetzen. Erst wenn Datenbanken detailliert behandelt wurden, also am Ende des zweiten Kurshalbjahres der Stufe 12, sollte ein im Unterricht erstelltes CMS mit Internetdatenbanken arbeiten.

2.3.2 Kurshalbjahr 11.2

Im zweiten Halbjahr der Stufe 11 werden die Schüler mit den Grundlagen des Programmierens vertraut gemacht. Im Laufe dieses Kurses kann das auf Frames basierende CMS auf einfache Weise durch einige Programm-Zeilen (Skriptsprache oder Server Side Includes) dergestalt verändert werden, dass die Navigationselemente automatisch erzeugt werden.

Dabei wird es nötig, sich dem Thema Sicherheit zuzuwenden, denn die hier benötigten Programme werden auf einem Server ausgeführt. Dessen Sicherheit hängt von der umsichtigen Programmierung der Skripte und sachgerechten Zuweisung von Dateizugriffsrechten ab.

2.3.3 Kurshalbjahr 12.2

Die Stufe 12 ist für die Entwicklung des Systems besonders wichtig. Mit den Kenntnissen über Datenbanken verfügen die Schüler über das nötige Rüstzeug, um ein CMS zu erstellen, das mit denselben Mitteln arbeitet wie professionelle Systeme. Dass der Umfang eines Schulprojektes allerdings keine professionellen Ausmaße annehmen kann, ist evident.

3. Praktische Umsetzung

3.1 „CMS“ mit Frames.

Frames unterteilen eine HTML-Seite in Bereiche, die unterschiedliche Dokumente oder unterschiedliche Bereiche eines Dokuments zeigen.

Durch den Einsatz von Frames kann man erreichen, dass bestimmte Bereiche einer Webpräsenz immer gleich aussehen (Kopfbereich, Fußbereich, Navigationsbereich) und nicht in jeder Datei notiert werden müssen. Sie erleichtern somit das Erstellen und die Pflege einer Website. Eine Webpräsenz mit Frames wollen wir daher bereits als CMS bezeichnen, wohl wissend, dass dies den Widerspruch einschlägiger Anbieter provozieren würde, wüssten Sie es denn.

Ein CMS mit Frames zu erstellen, ist nicht kompliziert.

Ein HTML-Dokument (`index.html`) enthält das Layout der Seite. In ihm wird die Position und Größe der Frames angegeben, aus denen die Seite bestehen soll. Zusätzlich wird eine Startseite für jeden Frame angegeben.

`Datei: index.html`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>CMS mit Frames</title>
</head>
<frameset rows="130,*,30" frameborder="0">
  <frame src="oben.html">

    <frameset  cols="20%,*,20%">
      <frame src="links.html">
      <frame src="mitte.html" name="content">
      <frame src="rechts.html" name="rechts">
    </frameset>

  <frame src="unten.html">
</frameset>
<noframes>
Ihr Browser unterst&uuml;tzt keine Frames!
</noframes>
</html>
```

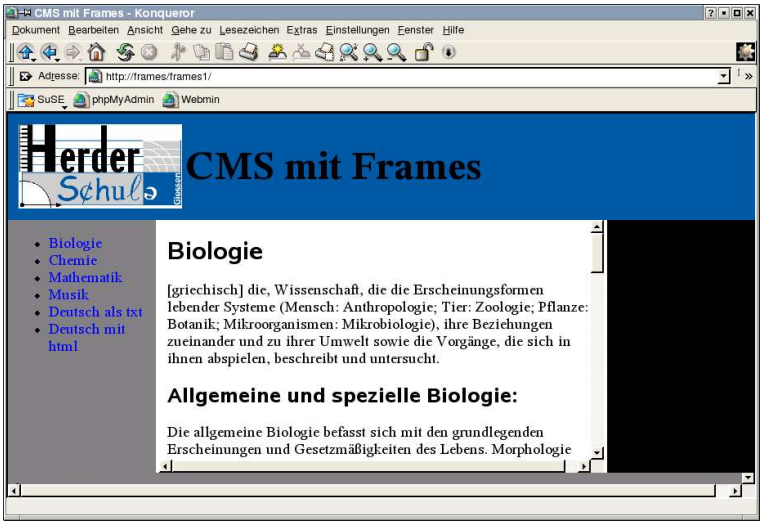
Zu beachten ist, dass am Beginn des Dokuments der Dokumententyp angegeben wird, der die Definition von Framesets zulässt. Zwar können Browser in der Regel auch ohne diese Angabe die Seite richtig darstellen, garantiert ist dies allerdings nicht. Gibt man den richtigen Dokumententyp an, sollten auch künftige Browser die Seite korrekt darstellen. Daher empfiehlt das W3-Consortium die Verwendung der Angabe[11].

Im DTD „Frameset“ darf der `<body>`-Tag nicht erscheinen. An seine Stelle tritt der Tag `<frameset>`.

11 http://www.w3.org/2001/06tips/Doctype

Um das Layout so zu gestalten, wie es in Bild 1 gezeigt ist, müssen zwei Framesets ineinander verschachtelt werden. Das äußere Set besteht aus drei Zeilen. Die zweite Zeile enthält das innere Set, das aus drei Spalten besteht.

Dimensionen der Frames lassen sich absolut in Pixeln oder in Prozent angeben. Ein Asterisk steht für automatische Breite bzw. Höhe.



Den `<noframes>`-Tag sollte man verwenden, wenn es auch kaum noch Browser gibt, die keine Frames darstellen können.

In `links.html` wird der Inhalt der linken Spalte definiert. Wichtig ist das Tag `<base target>`. Es legt fest, wo die Seiten geöffnet werden, auf die die relativen Links verweisen. Hier ist es der Frame `content`. In der Definition des Framesets wurde dieser Name der mittleren Spalte zugewiesen. Ohne diese Angabe würde der Navigationsbereich beim Anklicken eines Links durch die neue Seite ersetzt.

```
<html>
<head>
<base target="content">
<title>links</title>
<meta name="description" content="Frame links">
<link rel="stylesheet" href="common.css" type="text/css">
</head>
<body bgcolor="gray">
    <ul>
        <li><a href="biologie.html">Biologie</a></li>
        <li><a href="chemie.html">Chemie</a></li>
        <li><a href="mathematik.html">Mathematik</a></li>
        <li><a href="musik.html">Musik</a></li>
        <li><a href="text.txt">Deutsch als txt</a></li>
        <li><a href="text.html">Deutsch mit html</a></li>
    </ul>
</body>
</html>
```

Wird in diesem rudimentären CMS ein neuer Inhalt eingefügt, muss er lediglich als Textfile mit der Endung `.html` abgelegt werden. abschließend wird das Menu in `links.html` aktualisiert. An-

hand der Dateiendung entscheidet der Browser, wie der Inhalt dargestellt werden muss. Reiner Text wird völlig ohne Formatierungen angezeigt. Zeilenumbrüche richten sich nach der Breite des Darstellungsbereiches, also der mittleren Spalte. Möchte man Überschriften auszeichnen oder Absätze kenntlich machen, muss man die entsprechenden html-Tags manuell einsetzen.

Allerdings muss man sich nicht mehr um die Gestaltung des Kopf- und Fußbereichs kümmern. Durch das Frameset ist das Aussehen der Seite weitgehend festgelegt.

Auf dem beiliegenden Datenträger findet sich ein Beispiel unter `frames1`.

3.2 Erweiterung durch ein Submenu

Sollen nur wenige Seiten gepflegt werden, ist unser Einfach-CMS geeignet. Sobald aber die Zahl der Menupunkte so groß wird, dass die Übersichtlichkeit leidet, oder wenn Menüpunkte unter thematischen Gesichtspunkten gruppiert werden sollen, empfiehlt es sich, Submenüs zu erstellen. Dies ist mit der Frametechnik möglich.

In dem vorgeschlagenen Layout ist ein Frame (`rechts.html`) bereits für die Aufnahme des Submenüs vorgesehen. Es müssen nun zu jedem Oberbegriff Submenüs als HTML-File erstellt werden. Um die Submenüs in der rechten Spalte darzustellen, verfährt man auf die bereits bekannte Weise: Die rechte Spalte erhält in der Definition des Framesets einen Namen (hier: „rechts“). In der Datei für das Hauptmenu (`links.html`) wird dieser Name als Standardziel für Links (`base target`) angegeben. Die Submenüs verwenden ihrerseits die mittlere Spalte als `base target`.

Diese Variante ist unter `frames2` auf dem beiliegenden Datenträger zu finden. Im Unterschied zur ersten Version sind zusammengehörige Inhalte in Verzeichnissen abgelegt.

Soll in diesem System ein neuer Inhalt eingefügt werden, müssen unter Umständen die beiden Dateien angepasst werden, die Links enthalten. Der höhere Bedienkomfort wird durch höheren Wartungsaufwand erkauft.

3.3 CMS mit Server Side Includes

Mit einem sehr kurzen Programm lässt sich ein CMS erstellen, dessen äußeres Erscheinungsbild dem bisher dargestellten gleicht, welches aber wesentlich einfacher zu warten ist.

Der Inhalt der linken und rechten Spalte musste bisher bei jeder Änderung des Inhalts angepasst werden. Es wird nun eine Möglichkeit vorgestellt, diesen Prozess zu automatisieren. Neue Inhalte werden dazu als `index.html` bzw. `index.shtml` in einem Verzeichnis abgelegt. Der Name des Verzeichnisses wird als Link automatisch in den Navigationsbereichen dargestellt. Um diese Funktionalität zu implementieren, werden *Server Side Includes (SSI)* verwendet.

Will man diese Ausbaustufe als Projekt im Unterricht realisieren, müssen bereits erste Erfahrungen mit einer Programmiersprache vorhanden sein (Variablen, Schleifen, Bedingte Ausführung). Das zweite Halbjahr der Stufe 11 erscheint dafür angemessen.

3.3.1 Softwarevoraussetzungen

Bei den bisher beschriebenen CMS-Lösungen kann man das Ergebnis betrachten, indem man die Datei `index.html` mit einem Browser direkt vom lokalen Datenträger öffnet. Außer einem Texteditor zum Erstellen der Seiten sind keine weiteren Programme vonnöten. Ein Frameset kann daher ohne Schwierigkeiten von Schülern am heimischen PC erstellt werden. Der Einsatz von Server Side Includes setzt allerdings, wie die Bezeichnung schon nahe legt, den Betrieb eines Webservers voraus.

Wir gehen hier davon aus, dass für den Unterricht ein lokales Netz zur Verfügung steht, in dem auf einem Rechner (im weiteren Verlauf *Webserver* genannt) unter dem Betriebssystem Linux *Apache*[12] betrieben wird[13]. Dieser Webserver ist frei verfügbar und kann auf vielen Plattformen installiert werden. Auch in den angegebenen Beispielen wird von dieser Voraussetzung ausgegangen. Bei den meisten Linux-Distributionen ist Apache bereits vorkonfiguriert. Für Windows existiert ein komfortables Installationsprogramm.

Für das Experimentieren mit Webseiten muss jeder Schüler die Möglichkeit haben, seine HTML-Seiten auf dem Webserver zu publizieren. Daher sollte für Schüler der Zugriff auf ein Verzeichnis auf dem Webserver freigegeben sein.

Apache sucht HTML-Dateien in der so genannten `DOCUMENT_ROOT.` In der aktuellen Linux-Distribution von SuSE[14] ist dies `/srv/www/htdocs`. Man könnte dort für jeden Schüler ein Verzeichnis zum Schreiben und Lesen freigeben. Es gibt allerdings eine elegantere Methode.

Am einfachsten erstellt man für jeden Schüler einen Benutzer-Account auf dem Webserver. Für jeden Benutzer muss ein Verzeichnis mit dem Namen `public_html` im Homeverzeichnis existieren. Dort müssen die Schüler ihre Seiten via *ftp* oder auf anderem Wege ablegen. Der Name des Ordners ist konfigurierbar, `public_html` entspricht dem Standard. Der Lesezugriff für `others` muss sowohl für die Verzeichnisse als auch für die Dateien erlaubt sein, die Apache anzeigen soll.

Um die Seiten des Benutzers mit dem login `meier` auf dem Server `herder` anzusehen, steuert man mit dem Browser den URI `http://herder/~meier/` an. Apache sucht die Seiten nun nicht unter `DOCUMENT_ROOT`, sondern im Verzeichnis `/home/meier/public_html/.`

---

12 http://www.apache.org/

13 Für das Verständnis der folgenden Ausführungen sind zudem Kenntnisse über die Benutzerverwaltung und Zugriffsrechte unter Linux erforderlich.

14 Diese Distribution wird für die hier beschriebenen Programme zugrunde gelegt. SuSE-Linux ist unter http://www.suse.de erhältlich. Aktuell ist Version 8.2

Damit dieser Mechanismus funktioniert, muss das Modul *userdir* für Apache geladen sein. Bei den gängigen Linux-Distributionen ist dies in der Standardkonfiguration der Fall. Ansonsten kann man in der sehr informativen Dokumentation[15] des Apache nachlesen, wie dies genau bewerkstelligt wird. Eine detaillierte Beschreibung würde den Rahmen dieser Ausführungen sprengen.

Die Konfiguration des Apache wird in einer einfachen Textdatei festgelegt, die unter Linux in der Regel unter `/etc/httpd/httpd.conf` zu finden ist. Unter Windows sucht man sie in dem Verzeichnis, in dem Apache installiert ist. Änderungen werden, wie unter Linux üblich, mit einem Texteditor vorgenommen und sind nach dem Neustart des Programms wirksam.

Um Benutzerverzeichnisse zu aktivieren, muss die `httpd.conf` folgende Zeilen enthalten.

```
<IfModule mod_userdir.c>
   UserDir public_html
</IfModule>

<Directory /home/*/public_html>
   AllowOverride FileInfo AuthConfig Limit
   Options MultiViews Indexes SymLinksIfOwnerMatch Includes
   <Limit GET POST OPTIONS PROPFIND>
      Order allow,deny
      Allow from all
   </Limit>
   <LimitExcept GET POST OPTIONS PROPFIND>
      Order deny,allow
      Deny from all
   </LimitExcept>
   AddType text/html .shtml
   AddHandler server-parsed .shtml
</Directory>
```

Besonders das Schlüsselwort `Includes` ist wichtig, da es die Ausführung von SSI erst ermöglicht. Dasselbe gilt für die Zeilen, die mit `AddType` und `AddHandler` beginnen.

In der SuSE-Distribution sind Benutzerverzeichnisse aktiv und deren Konfiguration ist in eine eigene Datei ausgelagert.

3.3.2 Server Side Includes

SSI bieten eine einfache Möglichkeit, dynamische Inhalte zu bestehenden HTML-Seiten hinzuzufügen. SSI-Anweisungen werden dazu einfach in den HTML-Code eingefügt. Wird das Dokument an einen Client gesendet, werden die Anweisungen auf dem Server ausgeführt und durch ihre Ausgabe ersetzt. Für den Browser ist ihr Quelltext demzufolge nicht sichtbar. Der Browser muss daher in keiner Weise konfiguriert werden, wie das bei Java oder Javascript der Fall ist. Der Server hingegen muss für die Ausführung von SSI konfiguriert werden. Unterlässt man dies, werden Seiten lediglich ohne die Ausgabe der SSI dargestellt. Der Code erscheint ebenso wenig im Browser, da er stets durch HTML-Kommentarzeichen (`<!-- Kommentar -->`) geklammert ist. Er ist dann allerdings im Quelltext der Seite sichtbar.

SSI existieren bereits für den HTTPd - Webserver der NCSA[16], der in der Pionierzeit des WWW als Standard-Server fungierte. Der HTTPd kennt nur wenige Kommandos (vgl. nachstehende Tabel-

15 http://httpd.apache.org/docs/
16 http://hoohoo.ncsa.uiuc.edu/

le), während sein Nachfolger Apache mehr SSI-Direktiven versteht. Apache erlaubt beispielsweise Kontrollstrukturen (`if elif, else endif`).

Die Syntax für SSI-Anweisungen ist stets

```
<!--#Kommando parameter=wert parameter=wert ... -->
```

Hier eine Auflistung der wichtigsten Kommandos und ihrer Parameter:

Kommando	***Parameter***	***Beschreibung***
config	errmsg sizefmt timefmt	Einstellung des Ausgabeformats verschiedener Variablen (vgl. folgende Tabelle)
echo	var	Ausgabe von Umgebungsvariablen und speziellen SSI-Variablen (vgl. nachstehende Tabelle)
exec	cmd cgi	Ausführen von Programmen oder cgi-Skripten
flastmod	file	Ausgabe des Datums der letzten Änderung einer Datei
fsize	file	Ausgabe der Größe einer Datei
include	file virtual	Fügt eine Datei ein. *file* erwartet eine Pfadangabe relativ zum aktuellen Verzeichnis; *virtual* erwartet einen virtuellen Pfad (vgl. DOCUMENT_URI)
printenv	-	Ausgabe aller Umgebungsvariablen

In den SSI-Direktiven stehen folgende Variablen zur Verfügung:

Variable	***Bedeutung***
DATE_GMT	Aktuelles Datum und Zeit in Greenwich Mean Time
DATE_LOCAL	Aktuelles lokales Datum und Zeit
DOCUMENT_NAME	Name der aktuellen Datei
DOCUMENT_URI	Virtueller Pfad, relativ zur Wurzel des Servers.
LAST_MODIFIED	Datum der letzten Änderung der aktuellen Datei
QUERY_STRING_UNESCAPED	Undekodierter Querystring

Die aktuelle Implementierung des Musterservers Jigsaw[17] des w3-Konsortiums bietet den größten Sprachumfang[18]. Hier sind sogar Schleifen und Datenbankzugriffe möglich.

Ein HTML-File, das SSI-Direktiven enthält, kann so aussehen:

```
<html>
<head>
<title>SSI-Test</title>
</head>
<body bgcolor="white">
<h1>Dies sind die speziellen SSI-Variablen:</h1>
```

17 http://www.w3.org/Jigsaw/
18 http://www.w3.org/Jigsaw/Doc/User/SSI.html

```
<p>DATE_GMT:
<!--#echo var="DATE_GMT" -->
<p>DATE_LOCAL:
<!--#echo var="DATE_LOCAL" -->
<p>DOCUMENT_NAME:
<!--#echo var="DOCUMENT_NAME" -->
<p>DOCUMENT_URI:
<!--#echo var="DOCUMENT_URI" -->
<p>LAST_MODIFIED:
<!--#echo var="LAST_MODIFIED" -->
<p>QUERY_STRING_UNESCAPED:
<!--#echo var="QUERY_STRING_UNESCAPED" -->
<hr>
<p>Hier wird das Programm whoami ausgefuehrt. Wer bin ich?  
<!--#exec cmd="whoami" -->
<hr>
<p>Im File README steht:
<!--#include file="README" -->
<br>Die Groesse von README ist <!--#fsize file="README"-->
<hr>
<h1>Umgebungsvariablen:</h1>
<p>printenv:
<pre>
<!--#printenv -->
</pre>
</body>
</html>
```

Apache untersucht nicht jedes HTML-File auf SSI-Direktiven, weil dies sehr rechenintensiv ist. Sollen SSI in einem HTML-File ausgeführt werden, muss es eine bestimmte Endung haben. Üblich ist `.shtml`, allerdings kann man dies in `httpd.conf` konfigurieren.

Werden SSI-Kommandos nachträglich in bestehende Seiten eingefügt, müsste der Dateiname geändert werden und entsprechend alle Links, die auf diese Seite verweisen. Um diesen Aufwand zu vermeiden, kann man in `httpd.conf` den so genannten *XbitHack* aktivieren. Apache untersucht dann alle Files, in deren Dateirechten ein Executable-Bit gesetzt ist, nach SSI-Direktiven. Läuft der Webserver unter Windows, wo keine Dateirechte nach Linux-Art existieren, **müssen** SSI-Dateien die Endung `.shtml` haben.

Für unser CMS-Projekt sind lediglich die Direktiven interessant, die das Ausführen von Programmen erlauben. Dies sind `exec` und `include`.

3.3.2.1 exec

Die exec-Direktive führt ein Programm aus und wird dann durch dessen Ausgabe ersetzt.

Ein Beispiel:

```
<!--#exec cmd="ls" -->
```

Fügt man diese Zeile in ein SHTML-File ein, das sich im Wurzelverzeichnis des Servers befindet, wird ein Listing des aktuellen Verzeichnisses ausgegeben. Das auszuführende Programm muss in der `path`-Variablen vorhanden sein. Ansonsten muss der vollständige Pfad angegeben werden.

Wird die Zeile in ein HTML-File eingefügt, das sich in einem Benutzerverzeichnis befindet (public_html), so bleibt der erwünschte Effekt aus. Apache führt nämlich Programme und `cgi`-Skripte[19] unter der Identität des Benutzers aus, der das jeweilige Verzeichnis besitzt. Der Benutzer, unter dessen Identität Apache läuft (häufig wwwrun), darf beliebige Programme ausführen, sofern er die Rechte dazu besitzt. Normale Benutzer hingegen dürfen keine Programme ausführen, die sich außerhalb des jeweiligen Publikationsverzeichnisses befinden. Außerdem muss das Programm dem Benutzer gehören und er muss als Einziger Schreibrechte dafür besitzen[20]. Die Entwickler von Apache wollen durch diese und weitere Einschränkungen[21] die Ausführung von Programmen durch SSI sicher gestalten.

Ein Nachteil dieser Vorsichtsmaßnahmen ist offenkundig: Sobald ein Programm in verschiedenen Verzeichnissen benötigt wird, muss es auch jeweils dort physisch vorhanden sein; ein bloßer symbolischer Link genügt nicht.

Verwendet man bei der exec-Direktive anstelle des Parameters `cmd` einen anderen, nämlich `cgi`, kann diese Klippe umschifft werden.

```
<!--#exec cgi="~USER_NAME/cgi-bin/myprogram"-->
```

Diese Zeile sorgt in jedem SHTML-File eines Benutzers dafür, dass das Skript `myprogram` ausgeführt wird, welches im cgi-Verzeichnis dieses Benutzers liegt. Hier wird der Benutzername der Environment-Variablen USER_NAME entnommen. Man kann an deren Stelle auch den aktuellen Benutzernamen verwenden.

Der Ort und Name des cgi-Verzeichnisses kann selbstverständlich konfiguriert werden. Um allen Schülern die Möglichkeit zu eröffnen, in ihrem public_html-Verzeichnis cgi-Programme auszuführen, fügen wir der Konfigurationsdatei des Servers folgende Anweisungen hinzu:

```
<Directory /home/*/public_html/cgi-bin>
   AllowOverride None
   OPtions ExecCGI -Includes
</Directory>
```

In der SuSE-Distribution sind diese Zeilen nicht vorhanden. Sie sollten dort in der Datei /etc/httpd/suse_public_html.conf eingefügt werden.

3.3.2.2 include

Leider ist es nicht möglich, beim Aufruf eines cgi-Programm durch `#exec` Parameter zu übergeben. Die Dokumentation von Apache empfiehlt daher, die `#include`-Direktive zu verwenden, um Programme auf dem Server auszuführen[22].

Mit dieser Direktive werden, wie der Name erahnen lässt, Dateien eingebunden. Dabei können zwei Parameter verwendet werden:

`file`: Hier bezeichnet der Wert einen Pfad relativ zum aktuell angezeigten Dokument. Der Pfad darf kein „`../`“ enthalten und darf nicht absolut sein. Das heißt, dass keine Dateien eingebunden werden können, die in der Verzeichnishierarchie höher angesiedelt sind als das aktuelle Dokument.

19 CGI bedeutet Common Gateway Interface und beschreibt einen Standard für die Kommunikation zwischen Servern und externen Programmen. Eine genaue Beschreibung in englischer Sprache findet man unter http://hoohoo.ncsa.uiuc.edu/docs/cgi/overview.html

20 wwwrun darf mehrzeilige Parameter für cmd angeben. Normale Benutzer können das nicht.

21 Vgl. http://httpd.apache.org/docs/suexec.html

22 Vgl. http://httpd.apache.org/docs/mod/mod_include.html#includevirtual

`virtual`: Hier bezeichnet der Wert einen URL relativ zur aktuell angezeigten Webpräsenz, der lediglich eine Pfadangabe enthält. Beginnt er nicht mit einem „`/`“, nimmt Apache an, dass er relativ zum aktuellen Dokument zu interpretieren ist.

Zeigt der URL auf ein cgi-Programm, wird das Programm ausgeführt und seine Ausgabe an die Stelle der Direktive gesetzt. Da der Wert des Parameters ein URL ist, kann er einen so genannten Query-String enthalten. Dieser wird nach einem Fragezeichen an den URL angehängt. Im cgi-Programm steht der Query-String dann als Variable $QUERY_STRING zur Verfügung.

Wenn auf einem Webserver ein cgi-Verzeichnis eingerichtet ist, das das Programm `myprogram.cgi` enthält, kann man von jedem SHTML-Dokument dieses Servers aus das Programm mit folgender Zeile aufrufen, wobei der String „beispielwert“ in der Umgebungsvariablen `$QUERY_STRING` enthalten ist:

```
<!--#include virtual="/cgi-bin/myprogram.cgi?beispielwert" -->
```

3.3.3 SSI im CMS

3.3.3.1 #exec cmd

In der zuletzt beschriebenen Ausbaustufe unseres CMS werden zusammengehörige Inhalte in Verzeichnissen organisiert. Zu jedem Verzeichnis gehört eine HTML-Datei mit einem Submenü. Wird eine neue Kategorie hinzugefügt, muss ein Verzeichnis mit Inhalten erstellt werden und das passende Submenü, welches seinerseits im Hauptmenu eingetragen wird.

Diese Aktionen lassen sich durch ein Shellskript automatisieren. Dieses muss lediglich im aktuellen Verzeichnis nach allen Unterverzeichnissen suchen und sie in Gestalt von HTML-Links auflisten.

Unter einem Shellskript versteht man eine Datei mit einer Folge von Befehlen, die der Kommandozeileninterpreter, die Shell, verstehen und ausführen bzw. interpretieren kann. DOS-Benutzer kennen solche Dateien unter dem Namen „Stapelverarbeitungsdatei“ oder „Batchfile“.

In der ersten Zeile eines Linux-Shellskripts steht nach einem so genannten „SheBang“, der Zeichenfolge „#!“ („Sharp“ und „Bang“), der Pfad zu einem Programm, das die folgenden Zeilen abarbeiten soll. Gemeinhin sieht diese Zeile so aus:

```
#!/bin/sh
```

`"sh"` ist ein Alias für `bash`, den Kommandozeileninterpreter, der von den meisten Linuxdistributionen verwendet wird (**G**NU **B**ourne-Again **SH**ell). In deren Bedienungsanleitung, dem „Manual“ (mit '#man bash' aufzurufen) kann man die Syntax der benötigten Befehle nachlesen. Anstelle der `bash` kann ein Skript auch von einem anderen Interpreter ausgeführt werden, z.B. von `perl` oder `Tcl`.

Für unser Vorhaben, das Auflisten des Verzeichnisinhalts, ist nur eine Schleifenkonstruktion (`for`) nötig. Dazu kommt eine Kontrollstruktur (`if`), denn es muss überprüft werden, ob ein Verzeichnis oder eine Datei gefunden wurde. Schließlich benötigt man einen Ausgabebefehl (`echo`), um den HTML-Code zu erzeugen. Im Kurshalbjahr 11.2 müssen diese Voraussetzungen erarbeitet worden sein.

Die Syntax der `for`-Schleife, wie sie die `bash` anbietet, unterschiedet sich etwas von der, die Schüler von Delphi, Java oder Pascal kennen:

```
for NAME [in WORDS ... ;] do COMMANDS; done
```

Diese Schleife führt COMMANDS für alle WORDS aus. Die Variable NAME nimmt dabei der Reihe nach alle Werte von WORDS an. Sie muss nicht deklariert werden. WORDS wird von der Shell zunächst als Dateiname interpretiert. Dabei werden so genannte Wildcards berücksichtigt.

Um alle Dateinamen im aktuellen Verzeichnis zu erhalten, setzt man einen Stern für WORDS. Um auf die Variable NAME zuzugreifen, schreibt man $NAME. Das folgende Skript gibt den Inhalt des aktuellen Verzeichnisses aus und demonstriert die Verwendung der `for`-Schleife:

```
#!/bin/sh
for file in *;
do
    echo $file;
done;
```

Um das Verzeichnislisting in einem HTML-Dokument anzeigen zu können, schreiben wir diese Zeilen in eine Datei namens `menussi` (von *Menü* und *SSI*) und versehen sie mit den Rechten `rwxr--r--` . Im HTML-Dokument muss lediglich die `exec`-Direktive stehen:

```
<!--#exec cmd="menussi"-->
```

Wird das Skript in dieser Form aus einem HTML-Dokument aufgerufen, stehen lediglich alle Verzeichniseinträge ohne Umbruch hintereinander. Um nur die Verzeichnisse als Links darzustellen, müssen einige Modifikationen vorgenommen werden:

- `test -d file` liefert true, wenn `file` ein Verzeichnis ist. Da nur Verzeichnisse angezeigt werden sollen, verwenden wir diese Abfrage.
- Um Anführungszeichen innerhalb eines Strings auszugeben, müssen diese mit einem vorangestellten Backslash notiert werden. Sie werden sonst als String-Ende interpretiert.
- Aus demselben Grund werden Dateinamen in Anführungszeichen eingeschlossen. Namen aus mehreren Wörtern werden sonst nicht als zusammengehörend erkannt.

Das fertige Skript sieht demzufolge so aus:

```
#!/bin/sh
echo "<ul>";
for file in *;
do
    if test -d "$file";
    then
        echo "<li><a href=\"$file/index.shtml\">$file</a></li>";
    fi;
done;
echo "</ul>";
```

Aufgrund der Beschränkungen der `#exec`-Direktive (s.o.) wird dieses Skript (`menussi`) in jedes Verzeichnis kopiert. Zusätzlich muss in jedem Verzeichnis eine Datei `index.shtml` existieren, in der das Skript aufgerufen wird. Auf diese Weise erhalten wir ein relativ leicht zu erweiterndes und zu änderndes „CMS“.

Datei `index.shtml`

```
<html>
<head>
<base target="content">
<title>Menu rechts</title>
<meta name="description" content="Menu rechts">
```

```
<link rel="stylesheet" href="/common.css" type="text/css">
</head>
<body bgcolor="#EEEEEE">
<!--#exec cmd="menussi" -->
<hr>
</body>
</html>
```

Zu beachten ist hier, dass als Standardziel für die Darstellung neuer Seiten der mittlere Frame gesetzt wird: `<base target="content">`.

Soll eine neue Seite eingefügt werden, muss ein Verzeichnis angelegt werden, in dem eine Datei `index.shtml` mit dem Inhalt angelegt wird. Die Hyperlinks werden dann automatisch erzeugt.

Mehrere Texte kann man hintereinander auf einer Seite darstellen, indem man `menussi` etwas modifiziert. Anstatt von Verzeichnissen werden Dateien aufgelistet (`test -f file`). Allerdings werden nicht deren Namen angezeigt, sondern ihr Inhalt ausgegeben. Dies geschieht mit dem Programm `cat`, das unter Linux gewöhnlich vorhanden ist.

```
#!/bin/sh
for file in *.txt;
  do
     if test -f "$file";
     then
         cat "$file";
      echo "<hr>";
     fi;
done;
```

Es liegt nahe, diese Erweiterung in `menussi` zu integrieren, so dass sowohl Verzeichnisse als auch Dateien im Menu dargestellt werden. Dabei müssen allerdings `index.shtml` und `menussi` ausgeblendet werden (Die logische UND-Verknüpfung der Bedingungen wird in Shellscripts mit && notiert).

Zudem muss die Endung der Dateien abgeschnitten werden. Letzteres kann das Programm `basename`. Es eliminiert aus einem vollständigen Dateinamen den Pfadanteil und, wenn angegeben, auch die Extension. Im Skript muss der Aufruf dieses Programmes in so genannte Backticks[23] eingeschlossen werden, damit der Variablen `file` nicht sein Aufruf, sondern die Ausgabe zugewiesen wird.

```
#!/bin/sh
echo "<ul>";
for file in *;
do
    if test -d "$file";
    then
        echo "<li><a href=\"$file\">$file</a></li>";
    else
        if test "$file" != "index.shtml" && test "$file" != menussi; then
            file=`basename "$file" .html `;   # Backticks!
            echo "<li><a href=\"$file\">$file</a></li>";
        fi;
    fi;
done;
```

23 Auf Tastaturen mit deutschem Layout rechts neben dem Fragezeichen.

```
echo "</ul>";
```

Die hier beschriebenen Programmzeilen finden im Beispiel `SSI-exec-cmd` Verwendung, das auf dem beiliegenden Datenträger vorhanden ist.

3.3.3.2 #include virtual

Ein Nachteile des eben beschriebenen Systems fallen ins Auge: Dasselbe Programm muss mehrfach auf dem Webserver vorhanden sein. Bei einer eventuellen Änderung müssen alle Instanzen individuell modifiziert werden.

Dies lässt sich ändern, indem man anstelle der `#exec`-Direktive `#include` verwendet. Damit können wir unser Skript `menussi` an einem zentralen Ort ablegen, im `cgi`-Verzeichnis.

Damit das Skript als CGI-Skript funktioniert, muss eine kleine Änderung vorgenommen werden. Jedes CGI-Skript muss einen Header senden, der zu erkennen gibt, welcher Art die Daten sind, die es ausgibt. In unserem Fall handelt es sich um Text, also muss der Header "Content-type: text/html" angeben. Nach der HTML-Spezifikation[24] muss ein Header durch zwei Zeilenvorschübe beendet werden. Das Skript beginnt demzufolge mit:

```
#!/bin/sh
echo "Content-type: text/html"
echo
echo
```

Die `index`-Datei muss weiterhin in jedem Verzeichnis existieren, um das Skript aufzurufen.

Wird das Skript `menussi` in `cgi-bin` abgelegt und dort aufgerufen, kann es nur den Inhalt dieses Verzeichnisses anzeigen; es muss also der Pfad zu dem aktuellen Verzeichnis als Parameter übergeben werden. Wie bereits oben beschrieben, erwartet `#include` einen URI (Uniform Resource Identifier). Dieser kann einen Querystring enthalten, der einem CGI-Skript in der Variablen $QUERY_STRING zur Verfügung steht.

Der Pfad zur aktuell vom Server ausgegebenen SHTML-Datei ist der Umgebungsvariablen $SCRIPT_FILENAME zu entnehmen. Der Name des Benutzers, in dessen `public_html`-Verzeichnis die Datei liegt, steht in $USER_NAME.

Die SSI-Anweisung, die in jeder index.shtml-Datei stehen muss, sieht demnach so aus:

```
<!--#include virtual="/~$USER_NAME/cgi-bin/menussi.cgi?$SCRIPT_FILENAME" -->
```

`menussi` muss aus $QUERY_STRING das Verzeichnis ermitteln, dessen Inhalt es auflisten soll. Dies leistet das Kommando `dirname`. Es liefert den Pfadanteil eines Dateinamens. Wie oben im Fall von `basename` muss auch hier der Aufruf dieses Programmes in Backticks eingeschlossen werden. Die Variable F wird nur der leichteren Lesbarkeit wegen verwendet.

```
#!/bin/sh
echo "Content-type: text/html"
echo
echo
echo "<ul>";
# Wegen langer Dateinamen muessen Variablen in Anfuehrungszeichen  eingeschlossen werden.
F=`dirname "$QUERY_STRING"`;    # Dateiname wird abgeschnitten
cd "$F";  # Wechsel in das aktuelle Verzeichnis.
for file in *;
do
```

24 http://www.ietf.org/rfc/rfc2616.txt

```
    if test -d "$file";
    then
        echo "<li><a href=\"$file/index.shtml\">$file</a></li>";
    fi;
done;
echo "</ul>";
```

In dieser Form ist das CMS zwar eleganter programmiert, weist aber immer noch einen Nachteil auf: Die Verzeichnistiefe ist beschränkt. Muss man einen Menüpunkt untergliedern, funktioniert das Skript nicht mehr wie vorgesehen. Neu hinzugefügte Verzeichnisse erscheinen im mittleren Rahmen.

Man kann dem abhelfen, indem man für die Darstellung der Verzeichnislisten immer denselben Rahmen verwendet. Ein Link zum jeweils übergeordneten Verzeichnis ist auch schnell eingefügt:

```
echo "<a href=\"../menu.shtml\" target=\"_self\">nach oben</a><p>";
```

Allerdings sieht man so nicht, wo man sich jeweils in der Verzeichnishierarchie befindet. Dies kann man nur lösen, indem man die Bestandteile des Querystrings jeweils als Link auf das entsprechende Elternverzeichnis auflistet.

Hierzu muss eine Schleifenkonstruktion verwendet werden. Da nicht bekannt ist, wie tief die Verzeichnishierarchie ist, kommt nur eine `while`-Schleife in Betracht. Die Shell (Bash) bietet diese mit folgender Syntax an:

```
while BEDINGUNG; do
    Schleifenrumpf;
done;
```

Die Abbruchbedingung für die Schleife kann leider nicht in allgemein gültiger Form angegeben werden, da sie vom Ort abhängt, wo die HTML-Dateien auf dem Server abgelegt sind.

Die Ergänzung für `menussi` besteht aus wenigen Zeilen. Der Query-String steht hier in der Variablen QS:

```
LEVEL="../menu.shtml";    # Link auf Elternverzeichnis
while test "$QS" != "/home/$USER_NAME/public_html/SSI-include-virtual2"; do # Pfad anpassen!
    QS=`dirname "$QS"`;     # Letztes Verzeichnis abschneiden (Backticks)
    LINK=`basename "$QS"`; # Neues letztes Verzeichnis als Link anzeigen
    echo "<a href=$LEVEL target=\"_self\">$LINK</a><br>";
    LEVEL="../"$LEVEL;        # Link auf Elternverzeichnis um eine Ebene erweitern.
done;
```

Auf dem Datenträger ist ein Beispiel unter `SSI-include-virtual2` vorhanden.

3.3.3.3 Von SSI zu PHP

Die Verwendung von SSI bieten zweifellos interessante Möglichkeiten für die dynamische Gestaltung von Webseiten. Weil man alle Programme verwenden kann, die das jeweilige Betriebssystem bietet und eigene cgi-Skripte oder cgi-Programme verwendet werden können, ist prinzipiell alles machbar. Besonders attraktiv ist die Eigenschaft der ServerSideIncludes, dass ihre Verwendung für den Besucher einer Website nicht erkennbar ist. Wenn man den Quelltext einer Seite ansieht, kann man nicht feststellen, ob er statisch ist oder dynamisch erzeugt wird. Werden per `include` Programme eingebunden, bleibt deren Pfad verborgen. Dadurch ist die Wahrscheinlichkeit, dass ein Angreifer durch Manipulation des Query-Strings einen Absturz verursacht oder sich Zugriff auf

Server-Dateien verschafft, geringer als im Falle von Skriptsprachen wie z.B. PHP, deren Verwendung durch die Dateiendung häufig offenkundig wird.[25]

Für die Realisierung komplexer Internetauftritte haben sich dennoch seit einigen Jahren Skriptsprachen durchgesetzt, die die Erstellung dynamischer Webseiten unterstützen (z.B. Perl) oder die speziell dafür entwickelt wurden (z.B. PHP).

PHP wurde 1994 von Rasmus Lerdorf (University of Toronto) als eine Sammlung von Skripten für seinen eigenen Webserver entwickelt (*Personal HomePage tools*. Zuweilen wird PHP auch als Akronym für *PHP Hypertext PreProcessor* interpretiert). Später wurde PHP in C programmiert und es bildete sich ein Entwicklungsteam. Mittlerweile ist PHP eine professionelle Skriptsprache für die Entwicklung von dynamischen Webseiten und hat im Vergleich zu vergleichbaren Sprachen die größte Entwicklergemeinde. Der Quellcode ist frei verfügbar[26] und kann für viele Plattformen kompiliert werden. Im Gegensatz zu plattformspezifischen CGI-Programmen kann man PHP-Skripte demzufolge weitgehend ohne Anpassungen unter verschiedenen Betriebssystemen verwenden.

PHP wird auf dem Server ausgeführt. Wie bei den SSI muss am Browser daher nichts konfiguriert werden. Da PHP als Modul von Apache existiert, muss zur Ausführung von PHP-Programmen kein eigener Prozess auf dem Server gestartet werden, wodurch sich gegenüber der Ausführung von CGI-Programmen Performance-Vorteile ergeben.

Was PHP für einen Webdesigner und auch für den Unterricht attraktiv macht, ist die Tatsache, dass der Programmcode direkt in eine HTML-Seite eingebunden wird[27]. Eine bestehende Seite kann damit schrittweise modifiziert werden, so dass sie dynamische Inhalte präsentiert. Für die Verwendung in der Schule spricht zudem, dass man in PHP objektorientiert programmieren kann, aber nicht muss. Je nach Kenntnisstand der Schüler können Projekte also unterschiedlich realisiert werden.

Wie bei den SSI beginnt und endet PHP-Code mit einem Skript-Delimiter (`<?php ... ?>`). Im Unterschied zur `#exec-cmd`-Direktive der SSI können aber dazwischen mehrere Programmzeilen stehen. Damit wird die Entwicklung sehr erleichtert, denn man kann in der Regel mit nur einer Datei arbeiten, wohingegen man bei den SSI oft eine HTML-Datei und zusätzlich verschiedene CGI-Skript bearbeiten muss, deren Zusammenwirken erst die gewünschte Seite erzeugt.

3.3.3.4 Anpassungen der PHP-Installation

PHP ist in gängigen Linuxdistributionen enthalten und wird mit Apache als Modul installiert. Eine Konfigurationsdatei befindet sich in `/etc/php.ini`.

Schüler, die in eigener Regie mit PHP experimentieren wollen, finden Webhoster, die kostenlosen Webspace mit PHP4 zur Verfügung stellen. Unter Umständen sind Skripte, die auf dem Schulserver funktionieren, dort allerdings nicht mehr lauffähig, weil der kommerzielle Anbieter sein PHP anders konfiguriert hat.

Insbesondere wird in der Regel der so genannte `safe_mode` aktiviert, mit dem die Sicherheit auf gemeinsam genutzten Servern gewährleistet werden soll. Beispielsweise wird die Ausführung von Programmen verhindert, die nicht in einem festgelegten Verzeichnis liegen. Außerdem wird verhindert, dass auf Dateien zugegriffen wird, die nicht dem Eigentümer des laufenden Skripts gehören. Ein Passwort-File, das `root` gehört, kann so nicht mehr mittels PHP gelesen werden.

---

25 Zwar kann man einen Server so konfigurieren, dass auch PHP-Code in HTML-Dateien ausgeführt wird, allerdings erfordert dies Zugriff auf die Konfigurationsdateien. Bei kommerziellen Webhostern hat der Kunde diese Möglichkeit in der Regel nicht.

26 http://www.php.net

27 Mit Perl ist dies nicht möglich.

Außer dem `safe_mode` wird häufig `register_globals` ausgeschaltet. Damit sind Umgebungsvariablen nicht mehr global verfügbar. Wird allerdings `track_vars` aktiviert, sind Environment-Variablen über Arrays zugänglich (in der Form `$_SERVER[VARIABLE]`). Sie können damit nicht mehr ohne weiteres durch Manipulation des Querystrings sozusagen in ein Skript injiziert werden. Welche Variablen zur Verfügung stehen, erschließt sich durch die Funktion `phpinfo();` Schreibt man folgende Zeile in eine php-Datei und steuert sie mit dem Browser an, werden alle relevanten Eigenschaften der aktuellen PHP-Version sowie die vordefinierten Variablen übersichtlich dargestellt.

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

Mehr Informationen über die Konfiguration von PHP findet man, auch in deutscher Sprache, auf der Homepage der Entwickler[28]

3.4 SSI ersetzt durch PHP

Das zuletzt beschriebene CMS lässt sich mit wenig Aufwand auch durch PHP realisieren. Man muss lediglich das Skript `menussi` umschreiben und gegebenenfalls in der `index`-Datei jeden Verzeichnisses, die nun `index.php` heißen muss, einbinden. Da die benutzten Shell-Befehle teilweise sogar unter demselben Namen in PHP existieren, ist dies nicht schwer. Fast identisch wie in der `bash` verhalten sich `echo`, `dirname` und `basename`.

Das Auflisten eines Verzeichnisses ist etwas komplizierter. PHP stellt eine Pseudoklasse `dir` zur Verfügung, die die Methode `read` kennt. Ihre Verwendung wird bei der Betrachtung des Beispiels unter `php0` auf dem Datenträger deutlich. In dem Beispiel wird zudem demonstriert, wie mit einfachen Mitteln die übergeordneten Verzeichnisse in der richtigen Reihenfolge ausgegeben werden können. Dazu wird ein Array verwendet, auf das mittels `push` zuerst alle Paare aus Name und Link abgelegt werden. Mit `pop` werden die Paare danach in umgekehrter Reihenfolge gelesen und ausgegeben.

Diese Datei (`index.php`) wird in jedes Verzeichnis kopiert (symbolische Links erfüllen denselben Zweck). Im linken Frame wird der Verzeichnisinhalte und die Verzeichnishierarchie in Form von Links dargestellt. Der rechte Frame ist hier nicht mehr nötig.

```
<html>
<head>
<title>Kind-Menu </title>
<meta name="description" content="Menu">
</head>
<body bgcolor="#EEEEEE">
<?php
$qs=dirname($_SERVER["SCRIPT_NAME"]); // Pfadbestandteil des aktuellen
                                        #Skripts. Zugriff über Array wegen SAFE_MODE
$cd=basename($qs);                          # Name des aktuellen Verzeichnisses wird unten benötigt.
$level="../menu.php";                   # Link auf Elternverzeichnis
$stack=array();                             # Array deklarieren für push und pop .s.u.
while ($qs != "/~user/php0") {              # Hier unbedingt Verzeichnisnamen anpassen!
   $qs=dirname ($qs);                   # Letztes Verzeichnis abschneiden
   $link= basename ($qs);               # Neues letztes Verzeichnis wird linkname
   array_push($stack, $link);
   array_push($stack, $level);
   $level="../".$level;                 # „Elternlink“ ergänzen
}
while ($stack) {                        # Solange der Stack nicht durchlaufen ist...
```

28 http://de.php.net/manual/de/index.php

```
    $level=array_pop($stack);                # Link und Level in umgekehrter Folge lesen..
    $link=array_pop($stack);
    echo "<a href=$level target=\"_self\">$link</a> >> "; //... und ausgeben
}
echo "$cd";                                  # Aktuelles Verzeichnis als letztes ohne Link ausgeben
echo "<ul>";
$d = dir("./");                              # Directory-(Pseudo)Objekt erzeugen
while($file=$d->read()){                         # Alle Einträge lesen
    if (is_dir($file)AND $file[0] != (".") ) {
                                             # Wenn es sich um ein Verzeichnis handelt, Link erzeugen.
                                             #   Das erste Zeichen im Namen darf kein Punkt sein.
        echo "<li><a href='$file/menu.php' target='_self'>$file</a></li>";
    }
    else {
          if (!ereg ($file, " menu.php mitte.html unten.html oben.html
                  rechts.html  logo.png") ) {        #  nicht alle Files berücksichtigen
              $linkname= basename ($file, ".html");  # Dateiendung html abschneiden
              echo "<li><a href='$file' target=\"content\"> - $linkname</a></li>";
        }
    }
}
echo "</ul>";
?>
<hr>
</body>
</html>
```

Anstatt den Code in jeder `index.php` mitzuführen, könnte man auch, wie es bei der SSI-Variante getan wurde, das Programm an einer Stelle halten und mittels eines `include`-Befehls einbinden:

```
include („Dateiname“);
```

Eine Beispielimplementierung ist als php0 auf dem Datenträger vorhanden.

3.4.1 Nachteile von Frames

Bei allen bisherigen Ausbaustufen des rudimentären CMS fiel ein Nachteil besonders auf: Bei der Wahl eines Menüpunkts wurde im rechten oder linken Frame neue Links sichtbar, der mittlere Rahmen jedoch wurde erst nach dem Klick auf einen Datei-Link aktualisiert. Diese negative Eigenschaft von Framesets kann man mit JavaScript, das Frames sozusagen fernsteuern kann, kompensieren, aber dies soll nicht Gegenstand dieser Ausführungen sein.

Frames wurden in den beschriebenen CMS benutzt, weil es auf dieser einfachen Ebene eine bequeme Methode ist, gleichbleibende Inhalte nicht auf jeder Seite neu schreiben zu müssen. Dass im Informatik-Lehrplan die Kenntnis der Frame-Technik als verbindlicher Unterrichtsinhalt gefordert wird, sei nur am rande erwähnt.

In der Praxis bringt die Verwendung von Frames Nachteile mit sich:

- Nicht alle Browser können Frames anzeigen (heute, im Jahr 2003, weniger relevant)
- Frames erfordern unter Umständen höhere Ladezeiten, da mehr Dateien hintereinander angefordert werden.
- Das Setzen von Lesezeichen und das Verlinken auf Frameseiten wird erschwert.

- Suchmaschinen indizieren nur die Index-Seite, nicht aber die Frames selbst, da die Index-Seite keine Links enthält.

3.5 Ein dateibasiertes CMS mit PHP

Hat man Skriptsprachen zur Verfügung, die Dateien einbinden können, ist man nicht mehr auf Frames angewiesen. Demgemäß soll das nächste Projekt auf Frames völlig verzichten. Um dennoch auf das Seitenlayout Einfluss nehmen zu können, werden Tabellen verwendet.

3.5.1 Eigenschaften

Die neue Ausbaustufe des CMS soll folgende Eigenschaften haben:

1. Inhalte werden als Textfiles per `ftp` in Verzeichnissen abgelegt. Eine Datenbank wird nicht verwendet. In jedem Verzeichnis liegt dieselbe `index`-Datei, von der die Textdateien eingebunden werden.
2. Lauffähigkeit ohne Anpassungen auf verschiedenen Servern und in verschiedenen Unterverzeichnissen.
3. Konfigurierbarkeit von Titel und anderen Metatags für jede Seite
4. Automatisches Erstellen eines Navigationsmenüs

3.5.2 Realisierung

Zu 1. Auch in den bisherigen Systeme wurden die Inhalte als Dateien abgelegt. Diese wurden jedoch direkt angezeigt und mussten daher im HTML-Format vorliegen. Als Link wurde der Dateiname verwendet, wodurch die Reihenfolge der Anzeige vorgegeben war. Dasselbe gilt für die Verzeichnisse. Will man die Reihenfolge selbst beeinflussen, muss der Dateiname entsprechend selbst vergeben werden können. Dann aber muss der String, der als Link verwendet werden soll, in der Datei stehen.

Wer durch Raten oder Experimentieren die Dateinamen der Textdateien herausfindet, kann durch manuelle Eingabe des URI diese Files unter Umgehung der `index.php` im Klartext anzeigen. Dies ist nicht immer wünschenswert, denn sie könnten Informationen enthalten, die für das Funktionieren einer Seite nötig sind, aber der Öffentlichkeit vorenthalten bleiben sollten, wie Passwörter für Datenbankzugriffe oder Programmcode, der absolute Pfade auf dem Server enthält.

Apache bietet einen Mechanismus an, der den Zugriff auf Dateien beschränkt. Durch Anlegen eines Files `.htaccess` kann man verzeichnisspezifische Konfigurationen festlegen. Insbesondere besteht die Möglichkeit, den Zugriff auf ein Verzeichnis oder auf bestimmte Dateien auf bestimmte Benutzer oder Benutzergruppen zu beschränken. Dazu muss `.htaccess` etwa folgendes enthalten:

```
AuthName "CMS Administrationsbereich"
AuthType Basic
AuthUserFile /home/franz/users
<Files *.txt>
require user egon
</Files>
```

Hier wird der Zugriff auf alle Dateien mit der Endung `txt` beschränkt. Legt man solch eine Datei auf der obersten Verzeichnisebene der Webpräsenz an, kann nur noch der Benutzer `egon` auf solche

Dateien zugreifen, sofern er vorher sein Passwort richtig eingegeben hat, welches in `/home/franz/users` stehen muss. Dies gilt auch für Unterverzeichnisse.

Alternativ könnte man die Textdateien in einem Unterverzeichnis ablegen und den Zugriff darauf pauschal sperren. Mit PHP sind die Dateien immer noch zu lesen, da die Skripte nicht über das HTTP-Protokoll darauf zugreifen, sondern sie direkt öffnen.

Da es hier nicht nötig ist, dass überhaupt jemand über das HTTP-Protokoll direkt auf die Textdateien zugreift, reicht es aus, eine .htaccess-Datei mit obigem Inhalt im obersten Verzeichnis des CMS zu platzieren. An anderer Stelle wird erläutert, wie man Benutzer und Passworte anlegt.

Zu 2. Der zweite Punkt ist nicht so trivial, wie es scheint. Will man mehrfach verwendete Programmabschnitte per `include` einbinden, hängt der Pfad von der Verzeichnistiefe ab, in der sich das aktuelle Skript befindet. Ändert diese sich später, weil die HTML-Dateien an einen anderen Ort verschoben werden, muss die Pfadangabe angepasst werden. Könnte man absolute Pfade verwenden, wäre dies nicht nötig. Absolute Pfade verbieten sich aber, weil auf unterschiedlichen Servern die Document Root, das Verzeichnis, wo sich die HTML-Dateien befinden, unterschiedlich sein kann. Ein Ausweg scheint die Verwendung der von Apache bereitgestellten Variablen `DOCUMENT_ROOT` zu sein. Was unter Document Root zu verstehen ist, wurde bereits weiter oben erläutert[29].

```
<?php include ($_SERVER[DOCUMENT_ROOT]/file.inc) ?>
```

Diese Zeile funktioniert zwar auf jedem Server, allerdings nicht mehr in Benutzerverzeichnissen. Da die Schüler aber in eigenen `public_html`-Verzeichnissen arbeiten, muss eine andere Lösung her.

Die folgende ist zwar nicht unfehlbar, sollte aber in der Regel funktionieren: Wenn ein Skript in einem Benutzerverzeichnis liegt, ist das zweite Zeichen des `REQUEST_URI` stets eine Tilde (vgl. Ausgabe von `phpinfo()`). Dann aber kann man, wenn man davon ausgeht, dass die Benutzerverzeichnisse nach der Konvention in Apache `public_html` heißen, den Beginn von `SCRIPT_FILENAME` bis einschließlich `public_html` als Document-Root annehmen. Ist das zweite Zeichen des `REQUEST_URI` keine Tilde, liegt das Skript in dem Bereich auf dem Server, der in der `httpd.conf` als `DocumentRoot` angegeben ist.

Um das CMS-System in einem Unterverzeichnis ablegen zu können, muss dessen Pfad in einer Datei angegeben werden, die in der nun bekannten DOCUMENT_ROOT liegen kann, damit sie aus allen Unterverzeichnissen heraus referenziert werden kann.

Vgl. hierzu die `index.php` in den Verzeichnissen php* auf dem Datenträger.

Zu 3. Im Frameset werden nur die Metatags der `index`-Seite verwendet. Metatags der Frames sind für den Browser nicht sichtbar. Um in dem neuen CMS die Tags auf jeder Seite individuell gestalten zu können, kann man sie in einer Konfigurationsdatei festlegen, die beim Konstruieren der Seite eingelesen wird.

In einer Datei `siteconfig` werden zeilenweise die Konfigurationsdaten angegeben. Diese Datei wird in ein Array eingelesen, so dass jedes Element eine Zeile enthält. (Funktion `file`). Jede Zeile wird ihrerseits in ein Array überführt (Funktion `split`), wobei das erste Element das erste Wort ist und das zweite den Zeilenrest aufnimmt. Schließlich wird eine Variable mit dem Namen des Zeilenbeginns erzeugt, die den Zeilenrest enthält. (Ein solches Verfahren ist nur mit einer Skriptsprache möglich.) Zeilen, die mit einem # beginnen werden als Kommentarzeilen behandelt und daher ignoriert.

29 Vgl. Kapitel 3.3.1

In jedem Verzeichnis wird nach eine weiteren lokalen Konfigurationsdatei `config` gesucht, die, wenn sie existiert, ebenfalls verarbeitet wird und Standardeinstellungen überschrieben kann.

Zu 4. Hier kann man dasselbe Verfahren verwenden wie bei `php0`. Das aktuelle Verzeichnis wird nach Unterverzeichnissen durchsucht. Wenn im jeweiligen Unterverzeichnis ein File `title` existiert, wird dessen Inhalt als Linkname verwendet. Es wird nicht mehr, wie bei `php0`, jedes Verzeichnis aufgelistet. Der Verzeichnisname spielt keine Rolle mehr.

Links auf übergeordnete Verzeichnisse werden durch Zerlegen der Umgebungsvariablen `SCRIPT_NAME` erzeugt, bis der in der erwähnten ini-Datei angegebene Pfad bleibt.

Links auf die Textdateien im jeweils aktuellen Verzeichnis werden in der rechten Spalte angezeigt. Im Ordner `.files` werden die Namen aller Textfiles und der Inhalt ihrer ersten Zeile in einem so genannten Hasharray gespeichert. Dies ist ein Array, bei dem der Zugriff nicht über einen Index, sondern über einen Schlüsselstring erfolgt. In einem ersten Durchlauf durch das Array `text` werden die Dateien mittels `include` eingebunden und in der mittleren Spalte der Seite dargestellt, wobei die erste Zeile als Überschrift dargestellt wird. In einem zweiten Durchlauf werden die gefundenen Titel als Links auf die jeweilige Datei in der rechten Spalte dargestellt. Auch hier spielt der eigentliche Dateiname keine Rolle für den Link und kann daher als Sortierkriterium herhalten – `0index` wird über `3index` angezeigt.

Auf dem Datenträger findet sich eine Implementierung unter `php1`

3.5.3 Dateibasiertes CMS mit PHP und QUERY_STRING

Nachteilig ist noch, dass in jedem Verzeichnis eine `index`-Datei liegen muss, die nur den Zweck hat, anderen Programmcode einzubinden (`include/index.inc`). Eleganter wäre es, wenn man immer dieselbe `index.php` aufrufen könnte und ihr als Parameter das darzustellende Verzeichnis übergibt. Dies ist möglich, wenn man im URI dieses Verzeichnis als Query-String angibt. (etwa: `http://host/~user/php1/index.php?Ebene0/Ebene1/Ebene2`).

In dem Programmcode, der von jeder `index.php` eingebunden wird, sind dazu einige Änderungen nötig. Pfade, die relativ angegeben werden konnten, weil durch den URI das aktuelle Verzeichnis vorgegeben wurde, müssen nun absolut oder auf die Serverroot[30] bezogen sein. Dafür kann man die Information, in welchem Unterverzeichnis das CMS liegt, nun direkt in `index.php` angegeben oder zumindest die entsprechende Konfigurationsdatei (im Beispiel `php2.ini`) im selben Verzeichnis ablegen, da nicht mehr aus unterschiedlichen Verzeichnistiefen darauf zugegriffen werden muss.

Der Query-String steht für das php-Skript unter $_SERVER[QUERY_STRING] zur Verfügung. Zu Beginn wird er in eine Variable kopiert, die im Verlauf des Skripts verändert werden darf.

Dies muss z.B. geschehen, wenn mittels `dirname` die Links auf übergeordnete Verzeichnisse erzeugt werden.

Um Links auf die Textfiles in `.files` zu erhalten, muss die Variable erneut mit dem QUERY_STRING belegt werden.

Probleme entstehen immer dann, wenn Pfade aus einzelnen Bestandteilen, hier `$serverroot` oder `$docroot` und `$current_dir` sowie `$filename` zusammengesetzt werden. Unter Umstän-

30 Wir verstehen hier im Unterschied zur Apachekonfiguration unter Serverroot das oberste Verzeichnis auf dem Webserver, wie es sich für den Browser darstellt, also der URI ohne Pfadbestandteile, z.B. http://server.de/
In den Programmbeispielen bezeichnet die variable $serverroot dasselbe. Serverroot und Documentroot sind nicht identisch.

den entstehen Pfade mit aufeinander folgenden Slashes (home/user/php2//file). Dies kann auch geschehen, wenn der URI manuell angegeben wird. Im Programm sollte man solche Fälle berücksichtigen und beispielsweise am Beginn bereits sicherstellen, dass der QUERY_STRING immer mit einem Slash beginnt. Im weiteren Verlauf des Skripts kann man sich danach richten.

Wie die den QUERY_STRING verwendende Variante realisiert werden kann, erschließt sich am ehesten durch einen Vergleich von include/index.inc in /php1 und /php2 auf dem Datenträger.

Aufgabenstellung

Verändern Sie das CM-System so, dass nur noch eine index.php nötig ist. Dazu muss der Pfad zu dem anzuzeigenden Verzeichnis im Query-String übergeben werden.

Query-String nennt man eine Zeichenkette, die man nach einem Fragezeichen an den URI anhängt (in dem URI http://server/dir/index.html?abcdefg ist abcdefg der Query-String). In einem CGI-Programm und auch in php-Skripten steht der Query-String als Variable zur Verfügung.

In PHP ist es $_SERVER[QUERY_STRING].

3.5.4 Dateibasiertes CMS mit PHP und QUERY_STRING und Editor

Es ist auf Dauer recht unbequem, wenn man jeden neuen Inhalt erst in ein Textfile schreiben muss, welches mit ftp in ein Zielverzeichnis kopiert werden muss. Kleine Änderungen erfordern bereits relativ hohen Aufwand. Professionelle CM-Systeme bieten daher in der Regel einen Editor, der das Verändern der Seiteninhalte direkt auf dem Server erlaubt. Teilweise werden sogar WYSIWYG-Editoren angeboten.

Unser bestehendes php2 können wir relativ leicht um einen einfachen Editor ergänzen. Dazu wird ein Textfile in ein Formular geladen und danach wieder gespeichert.

Diese Funktionalität wird von verschiedenen Skripten bereitgestellt.

edit.php öffnet die Datei, die ihm per Query-String angegeben wird. Es erzeugt ein HTML-Formular (TEXTAREA), in dem der Inhalt der Datei angezeigt wird. Das action-Attribut des <FORM>-Tags wird mit einem weitern Skript namens save.php belegt. Drückt man auf den Submit-Button des Formulars, wird save.php aufgerufen. Dieses erhält im Query-String den editierten Text und speichert ihn in der Datei, deren Name ebenfalls im Querystring steht. Verwendet man HTML-Formulare, entsteht der Query-String automatisch in der Form (var1=wert1&var2=wert2...).

In PHP kann man ein einem Skript auf die so übergebenen Variablen direkt unter deren Namen zugreifen, beispielsweise durch <?php echo $var1 ?>.

Bei ausgeschaltetem register_globals stehen die Variablen in vordefinierten Arrays zur Verfügung: $_GET[var1].

Beim Entwurf des Formulars hat man die Wahl zwischen zwei HTTP-Methoden: GET und POST.

Mit GET wird der Query-String im URI an das aufgerufene Skript übergeben. Mit POST wird er an den HTTP-Header angehängt. Da die Länge des URI einschließlich Query-String bei GET auf 1024 Zeichen beschränkt ist, empfiehlt für das Übermitteln des Textes sich die Methode POST.

Damit Textdateien in das Editorformular geladen werden können, muss ein entsprechender Link angeboten werden. Edit

Er wird am besten direkt unter dem bereits vorhandenen Link angebracht, also in `include/index.inc` eingetragen.

Der relevante Abschnitt in `edit.php` kann folgende Form haben:

```
$fcontents = join (", file ($docroot.$_SERVER['QUERY_STRING']));
echo "
    <form action='save.php' method='POST'>
    <input type='hidden' name='filename' value='$_SERVER[QUERY_STRING]'>
    <textarea name='text' COLS='80' ROWS='24'>";
print (htmlentities($fcontents));
echo"</textarea><p><input type='submit' name='save' value='Save'></form>";
```

`htmlentities` konvertiert Umlaute und Sonderzeichen in HTML-konformes Format.

In `save.php` muss etwa stehen:

```
$fh=fopen($docroot.$_POST['filename'],"w");
if (!$fh){
    exit;
}
else {
    fwrite ($fh, stripslashes($_POST['text']));
}
```

`stripslashes` sorgt hier dafür, dass bestimmte Sonderzeichen nicht mit einem Backslash versehen werden.

Um deutlich zu machen, wie im Formular die Variablennamen entstehen, auf die in `save.php` zugegriffen werden kann, sind Variablennamen und die korrespondierenden Zuweisungen in dem Beispiel hervorgehoben.

Selbstverständlich muss der Benutzer, unter dessen Identität Apache läuft, Schreibrechte für die Textdateien haben.

Ist ein Text gespeichert worden, muss man manuell zur vorhergehenden Seite zurückgehen. Fügt man allerdings unter dem Speicherbefehl die Ausgabe eines HTTP-Headers ein, kann man erreichen, dass die editierte Seite sofort angezeigt wird:

```
$redirect=$serverroot."/?".dirname(dirname($_POST['filename']));
header ("Location: $redirect"); /* Weiterleitung */
```

Unter `php3` ist auf dem Datenträger eine Beispiel-Implementierung vorhanden.

3.5.4.1 Benutzer-Authentisierung

Die Möglichkeit, Texte zu editieren, steht in Beispiel `php3` jedem offen, was in den wenigsten Fällen wünschenswert ist[31].

Weiter oben wurden bereits `.htaccess`-Files erwähnt, die Apache veranlassen, den Zugriff auf bestimmte Seiten von einer Authentisierung abhängig zu machen. Dazu werden für das betreffende Verzeichnis in einem einfachen Fall folgende Direktiven angegeben:

```
AuthName "Zugang nur für authentisierte Benutzer"
```

31 Webseiten, die von Benutzern online editiert werden können, werden als WIKI bezeichnet und sind in großer Zahl im Internet zu finden. Vgl. http://c2.com/cgi/wiki

AuthType Basic
AuthUserFile */pfad/zu/einer/passwortdatei*
require user *benutzername*

Dies kann in der zentralen Konfigurationsdatei `httpd.conf` geschehen oder in einer Datei `.htaccess`, die in dem entsprechenden Verzeichnis angelegt wird. Für diesen Fall muss in `httpd.conf` für das Verzeichnis `AllowOverride AuthConfig` eingetragen sein.[32]

Benutzer und Passworte werden mittels eines Programms `htpasswd` erzeugt.

htpasswd /pfad/zu/einer/passwortdatei benutzername

Muss die Passwortdatei noch erzeugt werden, ist `htpasswd` mit der Option -c zu verwenden.

Wird ein Seite aus einem dergestalt geschützten Verzeichnis angefordert, sendet Apache im Header „HTTP/1.1 401 Authorization Required“, worauf der Browser einen Dialog zur Eingabe von Name und Passwort öffnet.

Hat man sich als berechtigter Benutzer ausgewiesen, steht eine weitere Umgebungsvariable REMOTE_USER zur Verfügung, die den angegebenen Benutzernamen enthält. In Abhängigkeit von REMOTE_USER kann unser CMS entscheiden, ob die Textdateien editierbar sein sollen oder nicht und einen entsprechenden Link nach `edit.php` anbieten. Dieser enthält im Query-String den Pfad zu dem Textfile, das editiert werden soll.

Soll nicht jeder Besucher eine Passwortabfrage durchlaufen müssen, kann man das Edit-Skript in ein eigenes Verzeichnis legen, welches mit `.htaccess` geschützt wird. Damit auch von dort der Zugriff auf alle Inhalte gewährleistet bleibt, wird eine `index.php` in das `admin`-Verzeichnis kopiert, die auch dafür Sorge trägt, dass nach Aufruf eines Links nicht wieder in die Serverroot gewechselt wird. Man erreicht diese, indem die Variable `$serverroot` um das Verzeichnis `admin` ergänzt wird. Die Konfigurationsdateien, die `index.php` in `etc` erwartet, können so leider nicht mehr gefunden werden. Man kann dem abhelfen, indem man das `etc`-Verzeichnis einfach nach `admin` kopiert. Ändert man dessen Inhalt, hat man gleichzeitig ein eigenes Layout für die Admininstrationsseite erzeugt.

Leider gibt es keine verlässliche Möglichkeit, einen einmal angemeldeten Benutzer wieder abzumelden. Ist REMOTE_USER erst einmal definiert, kann die Variable nicht mehr gelöscht werden. Der Browser sendet sie (bei Verwendung von `AuthType Basic` übrigens im Klartext!) mit jedem Seitenaufruf an den Server. Man kann allenfalls die Variable neu belegen. Dazu legt man unterhalb von `admin` ein weiteres Verzeichnis mit einer `.htaccess`-Datei an, die einen nichtexistenten Benutzer verlangt. In dem Passwortdialog gibt man beliebige Zeichen ein. REMOTE_USER ist damit neu definiert. Man wird die Meldung „Authorization Required“ erhalten. Den URI der Startseite muss man nun manuell eingeben, denn auf der Fehlermeldung des Servers kann man keinen Link platzieren.[33]

Will man lediglich die dem Administrator vorbehaltenen Links (Edit, Delete, Neu) nicht mehr sehen und die Seite aus der Perspektive eines normalen Benutzers betrachten können, hilft ein einfacher relativer Link („`../`“) auf das übergeordnete Verzeichnis.

Erweiterungsmöglichkeiten

Im Editor-Formular könnte eine Abbruchmöglichkeit durch einen Link auf die zuletzt besuchte Seite realisiert werden, oder indem man das Edit-Formular in einem neuen Fenster öffnet, das man

32 Vgl. Dokumentation http://httpd.apache.org/docs/howto/htaccess.html

33 Man könnte zwar mit PHP-Funktionen einen Passwortdialog erzwingen, nach dessen Abbruch ein Header gesendet werden könnte, der auf die vorhergehende Seite führt. Da wir aber eine komfortablere Authentisierungsmethode (Sessions) kennen lernen werden, wird hier auf eine detaillierte Darstellung verzichtet.

zum Abbrechen des Editiervorgangs einfach schließen kann. Außerdem kann man außer einem EDIT-Link eine Möglichkeit vorsehen, eine neue Textdatei anzulegen.

In der Beispielimplementierung `php4` sind diese Erweiterungen realisiert.

3.5.4.2 Exkurs: Überlegungen zur Sicherheit von PHP-Skripten

FIXME: Lehrplanbezug Sicherheit im Internet.

In `new.php` kann man sehen, wie ein Skript zum Anlegen eines neuen Textfiles aus `edit.php` entwickelt werden könnte. Da dieses Skript nur einen Weg aufzeigt, wurde auf aufwändige Sicherheitsüberprüfungen verzichtet. Man sollte beispielsweise sicher sein, dass ein Benutzer keine Pfade eingibt, die Verweise auf übergeordnete Verzeichnisse enthalten. Sinnvoll wäre, den Pfad aus dem Query-String zumindest in einem hidden-Feld zu übergeben und dem Benutzer nur zu erlauben, den Dateinamen einzugeben. Dazu müsste allerdings `save.php` entsprechend angepasst werden.

Auf der Basis von `edit.php` ist ein Skript `delete.php` zum Löschen von Textfiles leicht zu realisieren. Hier muss nur das Formular entfernt werden. Stattdessen wird ein Befehl zum Löschen der im Query-String angegebenen Textdatei eingesetzt.

Dass `delete.php` genau prüfen sollte, welche Datei im Query-String angegeben ist, liegt auf der Hand. Ohne eine solche Prüfung wäre es ein leichtes, das Skript aus einer fremden HTML-Datei heraus so aufzurufen:

```
http://server/~user/cms/admin/delete.php?/../../../geheimedaten.txt.
```

Dies setzt zwar voraus, dass der Angreifer das Passwort des Administrators kennt, allerdings gibt es keine Möglichkeit zu verhindern, dass ein Besucher des CMS solange Benutzer-Passwort-Kombinationen ausprobiert, bis er die gültige gefunden hat. Da diese Informationen ohnehin bei jedem Seitenaufruf eines authentifizierten Benutzers unverschlüsselt über das Netz übertragen werden, bedarf es unter Umständen nicht einmal einer solchen „Brute-Force“-Attacke. Ein so genannter Netzwerk-Sniffer braucht nur den Netzverkehr für eine gewisse Zeit zu überwachen, um die Zugangsdaten zu ermitteln.

Ist die Sicherheit unseres CMS in dieser Weise kompromittiert, kann ein Angreifer alle Dateien lesen, die für den Apache-Benutzer lesbar sind. Er muss dazu lediglich `edit.php` etwa auf folgende Weise aufrufen:

```
http://server/~user/php4/admin/edit.php?/../../../../etc/passwd
```

Hat man allerdings beim Zuweisen von Rechten für Dateien außerhalb der HTML-Verzeichnisse nicht die nötige Sorgfalt walten lassen und Schreibrechte für jedermann zugeteilt, können diese Dateien sogar gelöscht werden, indem man in obiger Zeile `edit.php` durch `delete.php` ersetzt.

```
http://server/~user/php4/admin/delete.php?/../../../../etc/httpd.conf
```

Etwas mehr Sicherheit lässt sich gewährleisten, indem man den Query-String auf Verweise in übergeordnete Verzeichnisse hin untersucht und adäquat reagiert, sollte die angegebene Datei sich nicht an einem Ort befinden, wo man sie erwartet.

Hilfreich ist dabei die Funktion `realpath()`, die alle symbolischen Links und Verweise der Form „`../`“ auflöst und den realen Pfad angibt. Ist dessen Beginn mit dem Beginn der Document-Root identisch, ist zumindest gewährleistet, dass keine Datei außerhalb dieses Bereichs betroffen ist.

3.5.4.3 Authentifizierung mit Sessions

HTTP ist ein verbindungsloses Protokoll. Ist ein PHP-Skript abgearbeitet, wurde allenfalls eine Ausgabe auf dem Bildschirm des Betrachters erzeugt. Das Programm ist beendet und ein Zugriff auf Variablen ist nicht mehr möglich. Ein erneuter Aufruf der Seite (Reload) startet das Programm von vorne. Zustände können daher von einem Aufruf des Skripts zum nächsten nicht ohne weiteres weitergegeben werden. Zwar können einem Skript Parameter auf dem Weg über den Query-String (GET) oder über den HTTP-Header (POST, s.o.) übergeben werden, so dass Variablen vorbelegt werden können, allerdings entstehen dadurch Sicherheitsprobleme, wie oben dargestellt wurde. Man kann ihnen nur begegnen, indem man in einem Skript zum einen die Herkunft des Aufrufers kontrolliert und darüber hinaus sicherstellt, dass übergebene Variablen bestimmten Anforderungen genügen.

In PHP existiert hierzu die Möglichkeit, Sessions zu definieren. Man bezeichnet damit ein Verfahren, Daten über mehrere Verbindungen hinweg zu erhalten. Damit können PHP-Skripte sich in ähnlicher Weise verhalten wie Programme mit einer Ereignisverwaltung.

Jedem Besucher einer Website wird eine (möglichst) einmalige Kennung, die Session-ID, zugewiesen. Sie wird in einem Cookie gespeichert oder bei jedem neuen Seitenaufruf im URI oder im HTTP-Header mitgeführt. PHP kann dann bei jedem Zugriff auf eine Seite prüfen, ob eine Session-ID existiert. Ist dies nicht der Fall, wird sie neu zugewiesen. Existiert sie bereits, werden die dieser Session-ID zugeordneten Variablen verwendet.

Anwendungen wie Online-Shops oder Kontenführung im Internet wären ohne Sessionmanagement nicht denkbar.

Um unser CMS mit einem Sessionmanagement auszustatten, sind nur wenige Änderungen nötig.

Am Beginn eines jeden Skriptes müssen einige zusätzliche Befehle ausgeführt werden, durch die überprüft wird, ob eine Session-ID existiert. Darauf werden gegebenenfalls Variablen belegt oder abhängig von ihrem Inhalt reagiert.

Mit `session_name('CMS');` legen wir fest, dass die Session den Namen CMS haben soll.

`session_save_path("$_SERVER[DOCUMENT_ROOT]/../tmp");` bestimmt, wo die zu bewahrenden Daten abgelegt werden. Es ist sinnvoll, hier einen Pfad zu wählen, der nicht unterhalb der Server-Root liegt, damit Sessiondaten nicht mit dem Browser angezeigt werden können[34].

`session_start();` Nach dieser Anweisung stehen die Sessionvariablen zur Verfügung, die in einem vorangegangenen Seitenaufruf erzeugt wurden. Nun ist es auch möglich, neue Sessionvariablen zu erzeugen und ihnen einen Wert zuzuweisen.

Das Array `$_SESSION` nimmt die Sessionvariablen auf. Man kann ihre Existenz mit `isset ($var)` überprüfen oder sie neu belegen (z.B. `$_SESSION[login]=admin`)

In `php5` wird der Benutzer auf eine Login-Seite umgeleitet, wenn er den Administrationslink anwählt. Hat er sich mit Name und Passwort authentisiert, wird die Sessionvariable `login` mit „admin“ belegt. In Abhängigkeit davon werden an anderen Stellen Edit-und Delete-Links angezeigt oder verborgen. Jedes Skript, das dem Administrator vorbehalten sein soll, wird um folgende Zeilen erweitert:

```
session_name('CMS');
session_save_path("$docroot/tmp"); // Hier liegen die Daten der Sessions
session_start();  // Be jetzt stehen die Variablen zur Verfuegung
```

34In der Beispielimplementierung `php5` wurde dies nicht beherzigt, damit die Installation sich einfacher gestaltet.

```
if ($_SESSION['login'] != "admin"){
    sleep (1); // Eine Sekunde warten...
    header ("Location: login.php"); // ... und dann zur Login-Seite umleiten
    exit;   // Dieses Skript auf jeden Fall beenden.
}
```

Im Gegensatz zur Authentifizierung durch Apache kann man die Sessionvariable login wieder löschen oder mit einem anderen Wert versehen, wodurch eine erneute Anmeldung nötig wird.

Im Beispiel wurden Name und Passwort direkt im Skript angegeben. Selbstverständlich ist es sinnvoller, derartige Daten an schwer zugänglichen Orten zu bewahren, wo sie nicht durch den Browser angezeigt werden können.

Mit dem Sessionmanagement ist es nicht mehr ohne weiteres möglich, Skripte direkt mit einem manuell eingegebenen URI aufzurufen. Eine erfolgreiche Attacke setzt voraus, dass der Angreifer die SessionID besitzt oder erraten kann. Dies ist relativ unwahrscheinlich, endgültige Sicherheit kann jedoch auch bei diesem Verfahren nicht garantiert werden.

`php5` erweitert `php4` um die Sessionfunktionen.

3.6 CMS mit Datenbank und PHP

Wiewohl das CMS in der vorliegenden Form recht ansprechend wirkt, muss konzediert werden, dass es professionellen Ansprüchen noch lange nicht genügt. Das Erzeugen neuer Textbeiträge ist zwar möglich, setzt aber die Kenntnis der Verzeichnisstruktur auf dem Webserver voraus. Dass der Apache-Benutzer Schreibrechte in den Verzeichnissen braucht, ist als gravierender Nachteil zu betrachten.

Nicht ohne Grund arbeiten daher kommerzielle CM-Systeme nicht auf Dateibasis, sondern verwenden Datenbanken.

Der Lehrplan Informatik des Landes Hessen sieht die Behandlung von Datenbanken für das zweite Halbjahr der Stufe 12 vor. An deren Ende, so wird vorgeschlagen, soll ein Projekt stehen. Hierfür eignet sich die Fortführung des CMS, das nunmehr seine Inhalte nicht mehr in Dateien, sondern in einer Datenbank ablegt.

Die Schüler müssen über grundlegende Fertigkeiten im Umgang mit SQL verfügen. Dass objektorientierte Programmierung im ersten Halbjahr Unterrichtsgegenstand war, kommt dem Vorhaben zupass, weil einige PHP-Funktionen, die für die Arbeit mit Datenbanken gebraucht werden, Objekte verwenden.

Zwischen Grundkurs und Leistungskurs kann man differenzieren, indem man die Anforderungen an das CMS flexibel gestaltet: Ein Grundkurs sollte in der Lage sein, Inhalte aus einer Datenbank auszulesen und in übersichtlicher Weise darzustellen. Für die Eingabe kann dann ein vorkonfektioniertes Editorskript verwendet werden, das mit jeder Datenbank arbeitet[35]. Um die Anforderungen zu reduzieren, wäre auch die Vorgabe eines ER-Modells denkbar.

Im Leistungskurs wird man schon auf Grund des umfangreicheren Zeitrahmens ein Autorenmodul selbst entwickeln und Zugangsbeschränkungen (Benutzerverwaltung) realisieren können. Das ER-Modell sollten LK-Schüler ebenfalls selbst entwickeln.

35 Ein solches Werkzeug ist als `dbbrowser` auf dem Datenträger bereitgestellt. Nach der Anpassung der Zugangsdaten erlaubt es INSERT, UPDATE und DELETE-Operationen auf beliebige MySQL-Datenbanken

3.6.1 Softwarevoraussetzungen:MySQL

In den letzten Jahren hat sich MySQL[36] als Quasi-Standard für Datenbanken im Zusammenspiel mit Webservern herauskristallisiert. Viele Anbieter von Webspace kombinieren Linux, Apache, MySQL und PHP und/oder Perl, so dass sich das Akronym LAMP für diese Kombination von Betriebssystem und Serversoftware etabliert hat[37].

MySQL ist für den nichtkommerziellen Einsatz frei verfügbar und dadurch für den Einsatz in der Schule prädestiniert. Die aktuelle Version von MySQL ist 4.xx, häufig wird aber noch 3.xx verwendet. Für das CMS-Projekt hat dies kaum Konsequenzen[38]. Die Datenbank ist in den gängigen Linuxdistributionen in der Version 3.23.xx enthalten, so dass auf die Installation hier nicht weiter eingegangen werden muss.

Vor dem Einsatz muss unter SuSE-Linux (Version 8.2) ein Superuser-Passwort festgelegt werden. (`mysqladmin -u root password neuespasswort`) Danach können neue Benutzer mit individuell unterschiedlichen Benutzerrechten angelegt werden. Die Datenbank besitzt eine eigene Benutzerverwaltung und ist von den Linux-Benutzerrechten völlig unabhängig. Es empfiehlt sich daher, für jeden Schüler eine Datenbank anzulegen und ihm ein eigenes Datenbankpasswort zuzuteilen. Die Privilegien der Benutzer sollten so gesetzt werden, dass jeder Schüler nur auf seine eigene Datenbank zugreifen kann.

Die Einrichtung der Datenbanken erledigt man am besten mit einem Webinterface namens `phpMyAdmin`[39] [40]. Es handelt sich auch hier um Open Source Software. `phpMyAdmin` wird auf dem Server in einem Verzeichnis unterhalb von DOCUMENT_ROOT installiert, z.B: in `myadmin`.

Die nötigen Einstellungen werden in einer Konfigurationsdatei `config.inc.php` vorgenommen. Detaillierte Informationen kann man in der phpMyAdmin beiliegenden Dokumentation entnehmen. Lediglich ein Fallstrick im Zusammenhang mit der Benutzereinrichtung soll erwähnt werden: Wird ein neuer Benutzer angelegt, muss angegeben werden, von welchen Hosts aus er auf die Datenbank zugreifen darf. Es reicht in der Regel nicht aus, „`%`“ anzugeben (jeder Host). Wenn MySQL und Apache auf demselben Rechner laufen, muss zusätzlich ein Benutzer mit gleichem Namen existieren, der von `localhost` zugreifen darf. Andernfalls wäre ein Anmelden mit phpMyAdmin nicht möglich.

3.6.2 Eigenschaften des CMS

Die Erstellung eines Pflichtenhefts, wie es in der professionellen Softwareentwicklung üblich ist, würde sehr viel Zeit beanspruchen und ist für die Realisierung eines Unterrichtsprojekts nicht unbedingt erforderlich. Allerdings sollte mit den Schülern im Vorfeld genau geklärt werden, welche Eigenschaften die von ihnen zu erstellende Software aufweisen soll. Ansonsten kann beim Entwickeln leicht das Ziel aus den Augen verloren werden.

Die Liste der Eigenschaften könnte so aussehen wie die folgende, die die Eigenschaften der Beispielimplementierung beschreibt:

- Das CMS ist auf einem Apache-Webserver mit PHP-Modul lauffähig.

36 http://www.mysql.com/
37 Da Apache, MySQL und PHP auch in Portierungen für Windows existieren, kann man auch ein WAMP-System konfigurieren.
38 Version 3 beherrscht noch keine UNION, was an einer Stelle etwas hinderlich ist.
39 http://www.phpmyadmin.net/
40 Es existiert auch ein Windowsprogramm, das für die Administration von Mysql verwendet werden kann. Allerdings ist dessen Entwicklung eingestellt. Vgl.: http://mysqlfront.venturemedia.de/

- Beiträge werden in einer MySQL-Datenbank gespeichert, die auf demselben Server läuft wie Apache.
- Beiträge können hierarchisch organisiert werden.
- Ein Navigationsmenu soll dynamisch erstellt werden.
- Das Layout wird von den bisherigen Projekten übernommen.
- Konfigurationsdetails werden in der Datenbank gespeichert.
- Eine Benutzerverwaltung gewährt unterschiedliche Rechte an Besucher, Autoren, Chefredakteure und Administratoren.
- Die Authentifizierung der Benutzer erfolgt mittels Sessionmanagement.
- Inhalte können nur von bestimmten Benutzern eingesehen werden können, darunter von einem anonymen Benutzer „gast“.
- Autoren verfassen Beiträge und können den Kreis der Leseberechtigten festlegen.
- Chefredakteure geben Beiträge nach Durchsicht frei oder weisen sie an den Autor zur nochmaligen Bearbeitung zurück.
- Administratoren haben alle Rechte. Insbesondere können sie Benutzer anlegen und löschen.
- Für Autoren gibt es ein Eingabeformular für Beiträge. Es erlaubt die Eingabe von Titel, Teasertext, Textinhalt und URI eines optionalen Bildes. Zudem kann darin festgelegt werden, wer den Beitrag lesen kann.

Aus den bereits vorhandenen Dateien `index.php` und `include/index.inc` kann man für das neue Projekt das Layout, also weitgehend den reinen HTML-Anteil, übernehmen. Alle Informationen allerdings, die in Dateien abgelegt waren, sollen nunmehr aus einer Datenbank gelesen werden.

Wird eine Datenbank verwendet, muss zunächst deren Design festgelegt werden. Dies geschieht in drei Schritten:

1. Erstellen eines Entity-Relationship-Modells
2. Transformation
3. Abschließende Normalisierung

3.6.3 ER-Modell

Für den Entwurf eines Entity-Relationship-Modells kann man auf der Basis der oben erwähnten Eigenschaften des CMS folgende Überlegungen anstellen:

In einem CMS werden **Artikel** veröffentlicht. Diese haben zumindest eine Überschrift und einen Textinhalt. Häufig sieht man auf Portalseiten einen so genannten „Teaser“, das ist in der Regel ein kurzer Hinweis auf den Textinhalt, häufig identisch mit den ersten Sätzen, die mit einem Hyperlink auf „mehr dazu...“ abgeschlossen werden. Eventuell gehört ein Bild zu dem Artikel. Artikel werden in der Regel zu Themenbereichen zusammengefasst. In einem CMS werden sie zu einem bestimmten Zeitpunkt veröffentlicht und nach einer gewissen Zeit wieder entfernt. Möglicherweise sind bestimmte Inhalte nur für einen beschränkten Leserkreis zugänglich.

Besucher der Webpräsenz lesen die Artikel. Gehen wir davon aus, dass sie sich registrieren können, um beispielsweise einen Newsletter regelmäßig zugestellt zu bekommen, oder um Artikel

kommentieren zu können, kennt das CMS von ihnen die Emailadresse und eventuell Name und Vorname sowie einen Loginnamen und ein Passwort.

Autoren schreiben die Artikel und lesen sie auch. Ein Autor hat Name und Vorname, Email-Adresse, einen Loginnamen und ein Passwort.

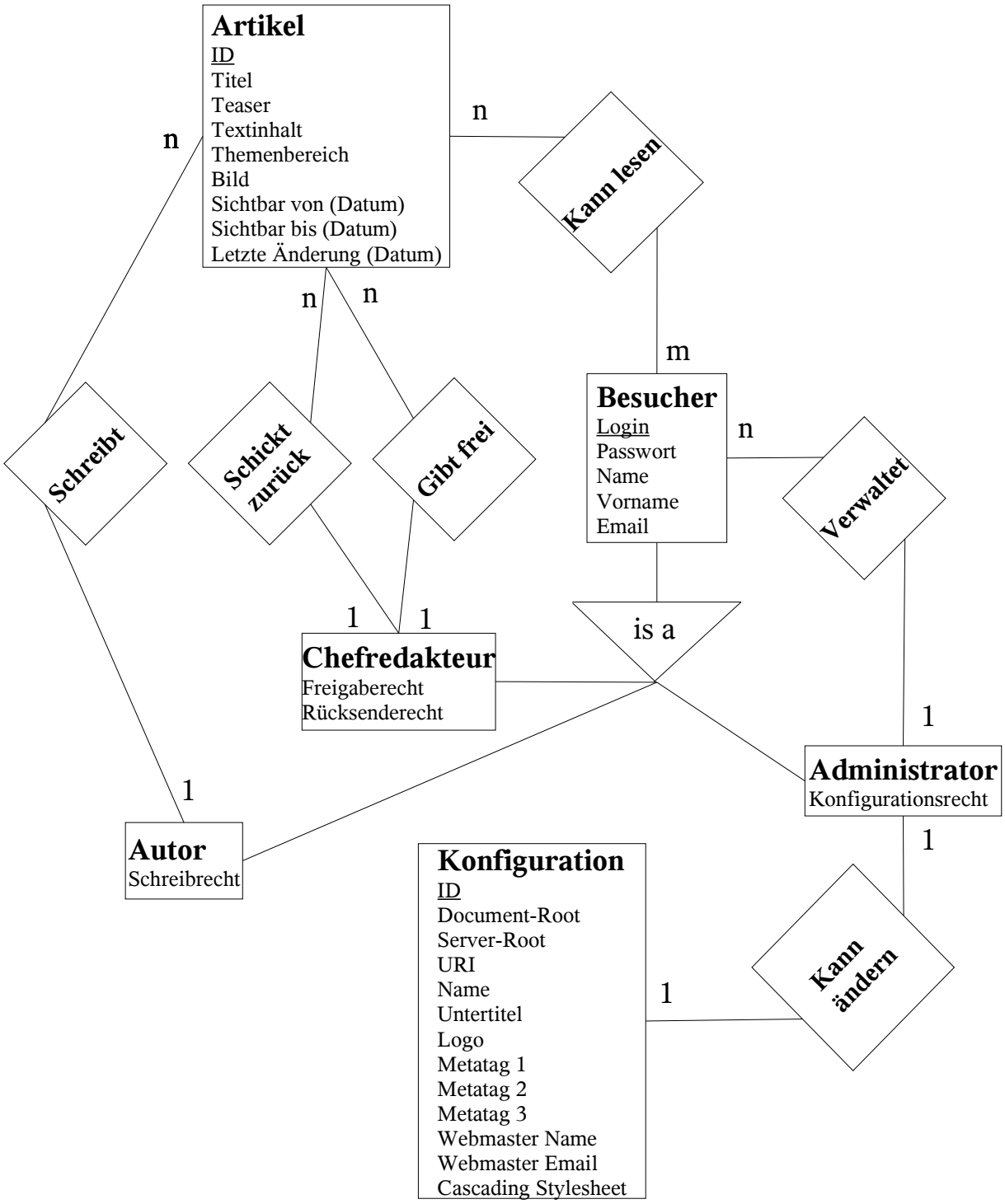
Ein **Chefredakteur** oder mehrere lesen die Artikel und geben sie zur Veröffentlichung frei. Gegebenenfalls schickt ein Chefredakteur einen Artikel an den Autor zur Überarbeitung zurück, bevor er veröffentlicht werden kann. Auch die Chefredakteure haben Name und Vorname, Email-Adresse, Loginnamen und Passwort.

Ein **Administrator** konfiguriert die Webpräsenz und verwaltet die Liste der Benutzer, Autoren und Chefredakteure.

Die Webpräsenz hat ein **Layout.** Dazu gehören alle Angaben, die bisher in einer globalen oder verzeichnisspezifischen Konfigurationsdatei gespeichert waren. Theoretisch realisierbar wäre ein individuelles Layout für jeden Artikel, wobei das Aussehen der gesamten Webpräsenz verändert werden könnte. In der Praxis wird man allenfalls Metatags ändern, wenn ein neuer Artikel angezeigt wird.

Als Schlüsselfeld eignet sich für die Entity *Artikel* keines der Attribute. Ein Titel könnte sich wiederholen, und auch die Kombination von Titel und Erstellungsdatum ist möglicherweise nicht einmalig. Da der Schlüssel zudem als Parameter im Query-String übergeben werden soll, führt man am besten einen künstlichen Schlüssel in Form einer Zahl ein. Für *Besucher* (einschließlich *Autoren* und *Chefredakteure*) kann der Loginname verwendet werden. Er darf nicht mehrfach auftreten und ist damit als Schlüssel geeignet. Für das Layout wird am besten ebenfalls ein künstlicher Schlüssel ID eingeführt.

Die grafische Darstellung dieses ER-Modells könnte etwa so aussehen:



3.6.4 Transformation

Das Entity-Relationship-Modell muss in Relationen transformiert werden.

Für die Transformation von Subtyp-Hierarchien wie wir sie bei *Autoren is a Besucher* haben, kann man mehrere Wege einschlagen:

1. Man kann für die Subtypen eigene Relationen einführen, die als Schlüssel den Primärschlüssel des übergeordneten Supertyps verwenden.
2. Man kann den Supertyp in eine eigene Relation überführen und die Attribute der Subtypen in diese Relation integrieren.

Da in unserem Fall die Subtypen nur wenige Attribute haben, sollte man diese in die Relation des Supertyps Besucher integrieren.

Besucher
Login
Passwort
Name
Vorname
Emailadresse
Freigaberecht
Schreibrecht
Rücksenderecht
Administrationsrecht

1:n-Beziehungen benötigen keine eigene Relation. Es genügt, den Primärschlüssel der Relation, die dem Entitytypen auf der 1-Seite entspricht, als Fremdschlüssel in die andere Relation aufzunehmen. Daher findet sich in *Besucher* bereits das Attribut *Administrationsrecht*. Da in der Praxis der Administrator, der die Benutzer verwaltet, derselbe sein wird, der die Konfiguration der Webpräsenz festlegt, führen wir kein eigenes Attribut *Konfigurationsrecht* ein.

Für die 1:n-Beziehungstypen *Schreibt*, *Schickt zurück* und *Gibt frei* wird ein Attribut in der Relation *Artikel* eingeführt, das den Loginnamen des Autors oder Chefredakteurs aufnimmt.

Artikel
ID
Titel
Teaser
Textinhalt
Themenbereich
Bild
Sichtbar von (Datum)
Sichtbar bis (Datum)
Letzte Änderung (Datum)
Autor
Zurückgeschickt von
Freigegeben von

n:m-Beziehungen erfordern eine eigene Relation. *Kann lesen* enthält demzufolge den Primärschlüssel der Relation *Artikel* und den von *Besucher*.

Kann lesen
Artikel.ID
Besucher.Login

Die Relation Layout braucht hier nicht extra aufgeführt zu werden. Sie erhält einen künstlichen Schlüssel, da alle Attribute mehrfach auftreten können.

Eine abschließende Normalisierung ist bei dieser Tabellenstruktur nicht mehr nötig.

Die auf dem Datenträger vorhandene Beispielimplementierung `myco` (MyContent) verwendet die hier angegebene Tabellenstruktur. Die im weiteren Verlauf angegebenen Beispiele beziehen sich auf `myco`.

In der Beispielimplementierung werden für die Attribute und Tabellen u.U. aus ästhetischen Gründen englische Bezeichner verwendet, die aber selbsterklärend sind.

3.6.5 MySQL-Zugriffe in PHP

Für den Zugriff auf eine Datenbank sind in der Regel vier Informationen nötig:

- Rechner, auf dem die Datenbank läuft (IP-Nummer oder Rechnername)
- Name der Datenbank
- Name des Benutzers
- Passwort des Benutzers

Damit diese sensiblen Daten nicht von einem Besucher versehentlich gelesen werden können, sollte man sie außerhalb der Serverroot ablegen und nur per `include` einbinden. Mit diesen Zugangsdaten wird vor jedem Zugriff auf die Datenbank zunächst eine Verbindung hergestellt. Da dies in einem CMS praktisch immer erforderlich ist, wird der nötige Programmcode unmittelbar angeschlossen.

Datei: `dbconnect.php`

```
<?php
define ("DB_HOST", 'localhost'); // Das CMS und MySQL residieren auf demselben Server.
define ("DB_NAME", 'cms');
define ("DB_USER", 'benutzername');
define ("DB_PASS", 'geheim');
$db = mysql_connect(DB_HOST, DB_USER, DB_PASS);
if(!$db) error("Keine DB-Verbindung");
mysql_select_db(DB_NAME, $db) or die("Zugriff auf ".DB_NAME." fehlgeschlagen!");
                                    // Ohne Datenbank sofort abbrechen.
?>
```

In diesem Codefragment werden zwei PHP-Funktionen verwendet: , `mysql_connect` zum Verbinden mit dem Datenbankserver und `mysql_select` für das Auswählen einer Datenbank. Dies kann nicht kombiniert werden, da es Benutzer geben kann, die Zugriffsrechte für mehr als eine Datenbank besitzen.

Die weiteren PHP-Funktionen im Zusammenhang mit MySQL sind in der Dokumentation zu PHP nachzulesen. An dieser Stelle sollen nur einige näher erläutert werden, die elementare Aufgaben erledigen:

1. `mysql_query(string sql)`; wird verwendet, um die SQL-Anweisung `sql` an die Datenbank abzusetzen. `sql` sollte nicht mit einem Semikolon enden. Die Verbindung zur Datenbank muss bereits in der oben beschriebenen Weise hergestellt sein. Für SELECT-Statements liefert der Befehl einen so genannten *Resource Identifier,* mit dessen Hilfe das Ergebnis der Abfrage gelesen werden kann. Für andere SQL-Befehle liefert `mysql_query` im Erfolgsfall TRUE oder ansonsten FALSE zurück. Selbstverständlich bedeutet TRUE nicht automatisch, dass auch Daten verändert wurden. Es heißt lediglich, dass die Abfrage an sich keine Fehler enthielt und damit vom Datenbankserver ausgeführt werden konnte.
2. `mysql_num_rows(resource Rersource-Identifier)` ermittelt für SELECT-Statements die Zahl der gefundenen Datensätze.
3. Um die bei einem SELECT-Statement gefundenen Daten zu verarbeiten, existieren mehrere Möglichkeiten. Sie werden in den folgenden Beispielen demonstriert, in denen jeweils aus der Tabelle *content* alle *text-F*elder aufgelistet werden:

Ermitteln der Anzahl der Datensätze, Auflistung mittels `for`-Schleife.

```
$query = "SELECT * FROM content WHERE ID=1";
$result = mysql_query($query);
if($result) { //Wenn ein Ergebnis existiert...
    $num_rows = mysql_num_rows($result);      // Zahl der Ergebniszeilen (Datensaetze)
    for($i=0; $i<$num_rows; $i++){
        echo mysql_result($result, $i, "text");
    }
}
```

Anlegen eines assoziativen Arrays, welches mittels `while`-Schleife durchlaufen wird.

```
$query = "SELECT * FROM content WHERE ID=1";
$result  =  mysql_query($query);
if ($result) {
  // Felder eines Datensatz in eine assoziatives Array überführen
  while ($ar=mysql_fetch_assoc($result)) {
    echo $ar['text'];
    }
}
```

Anlegen eines Arrays, durchwandern mittels `while`-Schleife. Das Text-Feld besitzt den Index 3.

```
$query = "SELECT * FROM content WHERE ID=1";
$result  =  mysql_query($query);
if ($result) {                  // Felder eines Datensatz in Array überführen
  while ($ar=mysql_fetch_row($result)) {
    echo $ar[3];                // ... und das 3. Element anzeigen.
    }
}
```

Erzeugen eines Objekts für jeden Datensatz. Das Attribut Text wird dargestellt.

```
$query = "SELECT * FROM content WHERE ID=1";
$result  =  mysql_query($query);
if ($result) {                  // Felder eines Datensatzes in Objekt überführen
```

```
    $ar=mysql_fetch_object($result);
    echo $ar->text;          // ... und Attribut text anzeigen.
}
```

Es wird deutlich, dass PHP eine sehr mächtige Sprache ist, die viele verschiedene Lösungsansätze ermöglicht. Jede der hier aufgeführten Vorgehensweisen bietet Vor- und Nachteile. Schüler, die an Pascal gewöhnt sind, werden die erste Variante bevorzugen. Dem bereits objektorientiert denkenden Schüler wird die letzte Variante eher liegen.

Die Wahl des passenden Befehls wird auch durch den jeweiligen Kontext bestimmt: Soll eine Tabelle vollständig ausgegeben werden, bietet sich `mysql_fetch_row` an; da man in dem für jeden Datensatz erzeugten Array jedes Element über seinen Index ansprechen kann, kann man mittels einer Schleife alle Felder leicht ausgeben. Die anderen Varianten eignen sich, wenn man Felder über deren Bezeichner ansprechen möchte.

Weitere Funktionen sollen hier nicht in aller Ausführlichkeit dargestellt werden. In der Dokumentation zu PHP sind alle Funktionen mit Beispielen aufgeführt.

3.6.6 Zugangskontrolle

Für die Authentifizierung wird in dieser Ausbaustufe unseres Projektes wieder das Sessionmanagement von PHP verwendet. Man kann es aus den vorangegangenen Projekten übernehmen, muss lediglich hier den Namen und das Passwort aus der Datenbank lesen. Eine Abfrage der Form
`SELECT * FROM user WHERE  log='$_POST[user]' AND pass = '$_POST[pass]'`
liefert nur dann ein Ergebnis, wenn die angegebene Kombination von Name und Passwort existiert. In diesem Fall wird der Benutzername für die Session als Login verwendet.

Abhängig von den Werten der Attribute *is_editor*, *is_author* und *is_admin* in der Tabelle *Besucher* kann man Links anbieten, die es erlauben, neue Texte zu schreiben, bestehende zu ändern, neue freizugeben etc.

3.6.7 Design

Das Aussehen der Website braucht sich von den bisher erarbeiteten Systemen nicht zu unterscheiden. Da allerdings mehr Funktionen implementiert werden sollen, muss man sich fragen, wo die Bedienelemente für diese Funktionen untergebracht werden können.

Prinzipiell wäre denkbar, dass man für die Autoren und Chefredakteure eine eigene Oberfläche programmiert. Obgleich ihr Design schmucklos sein kann, da es nicht für die Öffentlichkeit bestimmt ist, ist ihre Entwicklung mit Arbeit verbunden.

Etwas weniger aufwändig ist es, die Bedienelemente in das Layout zu integrieren und bei Bedarf anzuzeigen, wenn der aktuell angemeldete Besucher entsprechende Rechte besitzt. Der richtige Ort dafür ist jeweils der Bereich unter einem Artikel. Für Chefredakteure erscheinen dort beispielsweise die Links „Löschen“, „Freigeben“, „Zurück an den Autor“, „Erscheinungsdatum“ etc., für Autoren ist sichtbar „Editieren“, „Löschen“.

3.6.8 Anlegen der Datenbank

Vor dem Erzeugen der Tabellen muss geplant werden, welcher Spaltentyp für die jeweiligen Attribute verwendet werden soll. Da in einigen Tabellen Fremdschlüssel erscheinen, muss gewährleistet sein, dass sie gleichen Typs sind und die gleiche Länge haben, damit bei Zuweisungen keine

Zeichen abgeschnitten werden oder können. Zudem beeinflusst die Wahl des Datentyps den Speicherbedarf der Datenbank.

Für kurze Textfelder (Title, image, Pfadangaben, Namen) wird man den Typ `VARCHAR` mit seiner maximalen Länge von 255 Bytes verwenden. Im Gegensatz zu `TEXT` werden überflüssige Leerzeichen am Ende bei `VARCHAR` entfernt. Abfragen werden dadurch unter Umständen erleichtert, weil man nicht versehentlich eingegebene Leerzeichen berücksichtigen muss.

Für Felder, deren Länge 255 übersteigen kann, wie z.B. das Textfeld in *artikel*, eignet sich der Typ `TEXT`, dessen Maximallänge 65535 ist. Muss man befürchten, dass ein Artikel größer wird, kann man `MEDIUMTEXT` verwenden, worin man 1,6 MB speichern kann.

Der Primärschlüssel für *artikel* ist `INT`.

In der Tabelle *user* gibt es drei Felder (*is_author*, *is_editor*, *is_admin*), die nur zwei Werte aufnehmen sollen: Y oder N. Hierfür eignet sich der Typ `ENUM`. Er akzeptiert nur die Werte, die beim Anlegen des Feldes vorgegeben werden. Die Wahrscheinlichkeit von Fehlern, die durch Zuweisungen falscher Werte entstehen, wird dadurch verringert.

In der Tabelle a*rtikel* sollen die Zeitpunkte des Erstellens und Änderns festgehalten werden. Für diesen Zweck ist der Typ `TIMESTAMP` prädestiniert. Bei jedem Update-Vorgang wird in MySQL das erste Feld vom Typ `TIMESTAMP` automatisch auf das aktuelle Datum und die aktuelle Zeit gesetzt. Will man andere Timestamp-Felder derart aktualisieren, belegt man sie mit NULL. Diese Eigenschaft kann man sich zunutze machen, indem man als erstes `TIMESTAMP`-Feld das Änderungsdatum anlegt, das auf diese Weise immer das Datum der letzten Änderung enthält. Lediglich beim Einfügen eines neuen Datensatzes fügt man beim Erstellungsdatum, das in der Reihenfolge der Felder später kommen muss, NULL ein.

Ein SQL-Statement, das die Datenbank in der oben beschriebenen Form anlegt, findet sich auf dem Datenträger im Verzeichnis `myco`.

3.6.9 Navigationsmenü

In den vorangegangenen Projekten wurde das Navigationsmenü durch einfaches Auflisten von Verzeichnisinhalten erzeugt. Die Verzeichnishierarchie wurde durch iteratives Entfernen des jeweils letzten Pfadbestandteils ermittelt.

Verwendet man zur Speicherung der Inhalte eine Datenbank, sind diese Verfahren nicht mehr möglich. Um Inhalte zu gruppieren, muss ein anderes Ordnungskriterium entwickelt werden. In unserem ER-Modell ist das Attribut *Themenbereich* bereits vorgesehen. Damit ist es möglich, mehrere Kategorien anzugeben, zu denen ein Inhalt gehört, und ein zweistufiges Gliederungssystem zu realisieren.

Um weitere Gliederungsebenen hinzuzufügen, müssten weitere Attribute oder gar Entities in unserem Modell hinzugefügt werden. Da die Eigenschaften eines Themenbereichs sich aber von denen eines Inhalts kaum unterscheiden, gibt es eine einfache Methode, Gliederungsebenen in beliebiger Tiefe mit dem bereits existierenden ER-Modell zu realisieren. Dazu wird lediglich in dem Attribut Themenbereich der Primärschlüssel eines anderen Artikels angegeben. Auf diese Weise erhält man eine Baumstruktur, durch die das Löschen, Einfügen und Umgruppieren von Inhalten sehr einfach wird.

In dem nebenstehenden Beispiel ist in der ersten Klammer der Primärschlüssel des Beitrags und in der zweiten Klammer der jeweilige Vater-Schlüssel eingetragen. (0) bedeutet hierbei, dass es keinen Vater-Beitrag gibt und der entsprechende Menüpunkt in der obersten Ebene angesiedelt ist.

(1)-Thema1-(0)
    (4)-Unterthema1-(1)
    (5)-Unterthema2-(1)
(2)-Thema2-(0)
    (6)-Unterthema3-(2)
        (9)-Unterthema6-(6)
        (10)-Unterthema7-(6)
    (7)-Unterthema4-(2)
    (8)-Unterthema5-(2)
(3)-Thema3-(0)

Um ein Navigationsmenu zu erzeugen, wie man es von vielen professionellen Internetauftritten her kennt, kann man folgendermaßen vorgehen: Man selektiert zunächst alle Artikel, deren Vaterartikel *(0)* ist. Danach ermittelt man diejenigen, deren Vater der erste in der Liste ist. Auch diese Liste wird analog behandelt. Man erkennt unschwer, dass hier ein rekursives Vorgehen nahe liegt. Damit die Beiträge in der richtigen Reihenfolge in einem solchen Menü erscheinen, müssen sie auf einen Stack gelegt werden, der nach der Rekursion ausgegeben wird.

Will man nicht immer das vollständige Menü anzeigen, sondern nur den Ast, der den jeweils aktuellen Artikel enthält, muss eine entsprechende Bedingung eingefügt werden. Wenn im Beispiel Artikel 10 gewählt wurde, wird zwar Unterthema1 und Unterthema2 besucht, aber nicht ins Menü eingefügt.

In der Beispielimplementierung `myco` findet sich die Funktion `hierarchy`, die den hier beschriebenen Ansatz verwirklicht. (`myco/include/sys.php`). Weil der Codeausschnitt nicht sehr umfangreich ist, sei er hier wiedergegeben:

```
function hierarchy ($pid, $level, $indent, $limit) { // ID des Beitrags, Vaterbeitrag, Einrückung, Gefunden-
Flag (s.u.)
// Links werden in einem Array gesammelt. Beim Durchlauf durch das Array werden nur die Zweige
angezeigt,
// an denen das Blatt hängt, das aktuell gezeigt werden soll.

    $query = "SELECT DISTINCT * FROM  artikel WHERE parent='$level'";
    $result  =  mysql_query($query);// Die SQL-Abfrage
    if ($result) {
        $found=false;                   // Flag: Der aufgerufene Artikel ist noch nicht gefunden.
        $menu = array();                // Mehrdimensionales Array für die Menuzeilen.
        while ($ar=mysql_fetch_object($result)) { // Felder eines Datensatzes als Objekt
            if ($pid == $ar->ID){                   // Der angeklickte Link wurde gefunden.
                $found=True;                        // Jetzt muss noch eine Ebene "aufgeklappt" werden.
            }
                                                    // Gefundene Links werden an das Array gehängt.
            array_push($menu, array ("indent" => $indent, "ID" => $ar->ID,
                                        "title" => $ar->title, "show"=>$found));
                                    // In Abhängigkeit von dem letzten Element werden Menüpunkte
                                    //  an die der übergeordneten Ebene angehängt. (s.u.)

            if ($found){            // Blatt gefunden (s.oben). Jetzt nur noch eine Ebene weiter ausklappen,
                                    // um Navigation zu dessen untergeordneten Artikeln zu ermöglichen..
                                    // Dazu Parameter limit true setzen ($found)
                $t =   hierarchy($pid, $ar->ID, $indent."  ", True);          // Rekursion.
                $menu=array_merge($menu, $t); // Nur hier wird das von der Funktion
                // zurückgelieferte Array an das bereits vorhandene angehängt.
                $found=false;       // Damit weiter unten im Menü nichts mehr
                                    //ausgeklappt wird, Flag wieder umdrehen.
            }
            if (!$limit){ // Unterebenen sollen weiter durchsucht werden.
                $t =   hierarchy($pid, $ar->ID, $indent."  ", $found);        // Rekursion.
                foreach ($t as $line){      // Alle Elemente, die die letzte Rekursion geliefert hat,
                                    // nach einem show==true durchsuchen.
                    if ($line['show']) {
```

```
                    $menu=array_merge($menu, $t); // .. und nur dann diesen Zweig an
                                                  //  das Auszugebende anhängen.
                    break;
                }
            }
        }
    }
    return $menu;   // Alle gefundenen Menueintraege (Links) zurückliefern.
  }
}
```

3.6.10 Das Autorenmodul

Autoren sollen die Möglichkeit haben, neue Texte zu schreiben und solche, die noch nicht freigegeben oder vom Chefredakteur zurückgewiesen wurden, zu überarbeiten oder zu löschen.

Im Projekt `php5` wurden dafür verschiedene Skripte verwendet. Will man im Unterricht Aufgaben auf verschiedene Schüler verteilen, kann man dies auch hier so handhaben. Die Einzelergebnisse lassen sich später zu einer Datei zusammenfassen.

Eleganter ist allerdings eine Lösung, bei der alle Operationen von Beginn an in einem Skript vereinigt sind. Für die Entwicklung ist diese Vorgehensweise vorteilhaft, da man so leichter die Übersicht behält und nicht fortwährend zwischen mehreren Dateien wechseln muss.

Man kann dies realisieren, indem man für verschiedene Aktionen das Skript sich selbst mit unterschiedlichen Parametern aufrufen lässt. Der Programmcode für diesen Zweck kann allgemein formuliert werden, denn PHP bietet eine Umgebungsvariable, die den Namen des aufrufenden Skripts enthält: PHP_SELF.

In dem Skript werden Formulare definiert, deren Submit-Buttons sämtlich denselben Namen erhalten, z.B. `button`. Ihr Value, der in der Darstellung als Beschriftung erscheint, wird im Query-String übermittelt (button=*value*). Je nach Inhalt der Variablen $_POST[button] kann beim Aufruf des Skriptes reagiert werden. Die Kontrollstruktur `switch` eignet sich dafür besonders gut.

Das folgende Beispiel verdeutlicht die Vorgehensweise:

```
<?php
if(isset($_POST['button'])){   // Wenn dieses Skript überhaupt mit einem Button aufgerufen wurde...
    switch ($_POST['button']) {
        case 'Insert': // Aktuelle Werte als neuen Artikel einfügen.
            (Code hier nicht abgedruckt)
        break; //Insert
        case 'Update': // Artikel ändern.
            (Code hier nicht abgedruckt)
        break; //Insert
        case 'New': // Wird in der Artikeldarstellung der Button “New” gewählt...
            echo "<form action='$_SERVER[PHP_SELF]' method='POST'>";
            echo "<input type='hidden' name='author' value=$user->login>";
            echo "<input type='hidden' name='articleid' value=$articleid>";
            // Durch den Submit-Button Insert wird dieses Skript erneut aufgerufen.
            echo "<p><input type='submit' name='button' value='Insert'></form>";
        break; // new
        case 'Edit':
            echo "<form action='$_SERVER[PHP_SELF]' method='POST'>";
```

```
            ...
        echo "<input type='hidden' name='articleid' value=$articleid>";
        echo "<p><input type='submit' name='button' value='Update'
            onClick=\"return confirm('Datensatz aus artikel wirklich ändern?')\">
            <input type='submit' name='button' value='Insert'
            onClick=\"return confirm('Datensatz als neu einfügen?')\">";
        echo "</form>";
    break;
    default:
        echo "Noch nicht implementiert: $_POST[button]";
  } //switch
} // if isset(button)
```

Theoretisch wäre es denkbar, alle Funktionen in `index.php` unterzubringen. Praktische Erwägungen sprechen allerdings dagegen. Bei jedem Aufruf der Seite müsste auf dem Server `index.php` vollständig gelesen werden. Dass die an den Client übertragene Datenmenge unter Umständen viel geringer ist als die Dateigröße von `index.php`, liegt auf der Hand. Daher werden die in der Regel selten benötigten Funktionen in eine eigene Datei ausgelagert. Wird diese Datei (im Beispiel `edit.php`) aufgerufen, um beispielsweise einen Datensatz einzufügen oder zu ändern, wird nach erfolgreicher Ausführung nichts zu sehen sein.

Durch das Absenden eines HTTP-Headers kann man den Browser allerdings veranlassen, die Hauptseite wieder zu laden. Das Verfahren wurde bereits in `php3` verwendet.

3.6.11 Fallstricke

Beim Zusammenspiel von Datenbank (SQL) und PHP-Skript gibt es naturgemäß viele Fehlerquellen.

- Wenn ein Datenbank-Feld nicht belegt ist, heißt dies keineswegs, dass es leer ist. Manche Feldtypen nehmen beim Erzeugen der Tabelle einen vom Leerstring verschiedenen Standardwert an. Zudem ist NULL nicht gleichbedeutend mit einem leeren String. Bei SELECT-Abfragen muss dies berücksichtigt werden.
- Zuweilen muss in einem SQL-Statement ein Bezeichner in Hochkommata eingeschlossen werden, an anderen Stellen ist es nicht nötig. Das oben angegebene Statement

  ```
  SELECT * FROM user WHERE   login='benutzer' AND password='password'
  ```

  funktioniert nicht, wenn es so geschrieben wird:

  ```
  SELECT * FROM 'user' WHERE  login=benutzer AND password=password
  ```

  Auch Groß-Kleinschreibung spielt ein Rolle. Wenn eine SQL-Abfrage nicht das erwartete Ergebnis liefert, lohnt es sich, den Fehler auch in der Syntax zu suchen.

3.6.12 Datenbankoperationen

3.6.12.1 SELECT

Hat man die Datenbank wie beschrieben angelegt und mit einigen Einträgen bestückt, ist es ein Leichtes, Inhalte darzustellen. Dazu wird eine Abfrage an die Datenbank gestellt, die je nach angemeldetem Benutzer unterschiedliche Gestalt hat.

Es genügt während der Entwicklung des Projektes zunächst, die Abfragen bei der Erstellung des Navigationsmenüs (im Beispiel `include/sys.php`) zu verwenden, um die Einträge herauszufiltern, die ein Benutzer sehen darf. Damit ist es immer noch möglich, URIs manuell einzugeben und die Restriktionen zu umgehen, bis auch die Abfragen in `index.php` analog integriert werden. Danach kann ein Benutzer nur noch Artikel sehen, für die er tatsächlich autorisiert ist.

Benutzer mit **Administratoren**status und **Chefredakteure** müssen alle Artikel sehen können, deren ID im Querystring übergeben wird (`$article_id`), unabhängig von Veröffentlichungszeitraum, Veröffentlichungsfreigabe oder Leserecht. Ein entsprechendes SQL-Statement ist daher am einfachsten zu erstellen. Da in der Tabelle *artikel* nur der *login* des Autors steht, können wir seinen vollen Namen unter dem Artikel nur drucken, wenn wir ihn aus der Tabelle *user* lesen. Es findet daher ein JOIN statt:

```
SELECT * FROM artikel,user WHERE artikel.author=user.login
```

Für **anonyme Besucher** sollen Artikel nur zu sehen sein, wenn sie ein Chefredakteur freigegeben hat. *publishok_by* darf also nicht NULL sein. Weiterhin muss in *permission* ein Eintrag mit *artikel_id* und dem *login* des Anonymous stehen (gast).

```
SELECT * FROM artikel,user,permission WHERE
artikel.ID='$articleid' AND
publishok_by IS NOT NULL AND
artikel.author=user.login AND
artikel.ID=permission.ID AND
permission.login ='gast'
```

Für Besucher, die nicht anonym sind, dennoch aber weder Autor, Redakteur noch Administrator sind, ergibt sich eine analoge Abfrage. wobei statt gast eben der jeweilige *login* steht. Da ein individueller Besucher dem Standard-Gast gegenüber nicht benachteiligt sein soll, muss er auch für die Artikel Zugriffsrecht haben, für die er nicht explizit in *permissions* aufgeführt ist. Damit ergibt sich als Abfrage:

```
SELECT * FROM artikel,user,permission WHERE
artikel.ID='$articleid' AND
publishok_by IS NOT NULL AND
artikel.author=user.login AND
artikel.ID=permission.ID AND
(permission.login ='gast'  OR permission.login ='$user->login')
                    // Mit $user ist hier der aktuell angemeldete Benutzer gemeint, nicht die Tabelle user
```

MySQL wertet AND vor OR aus, wodurch es nötig wird, die letzten Bedingungen zu klammern.

Da die zweite Abfrage die erste umfasst, reicht es aus, nur sie für alle Besucher zu verwenden, die nicht Autor, Redakteur oder Administrator sind. Allerdings kann es nun sein, dass für einen Artikel zwei Ergebniszeilen zurückgeliefert werden; man muss also mittels `GROUP BY artikel.ID` dafür sorgen, dass nur eine berücksichtigt wird.

Autoren sollen alle Artikel sehen, die sie selbst geschrieben haben, unabhängig von der Freigabe durch den Chefredakteur. Das obige SQL-Statement muss also erweitert werden: *publishok_by* muss nicht NULL sein, es sei denn, der Artikel stammt von dem Besucher.

```
SELECT * FROM artikel,user,permission WHERE
artikel.ID='$article_id' AND
(publishok_by IS NOT NULL OR (publishok_by IS NULL AND artikel.author=$user->login)) AND
artikel.author=user.login AND
artikel.ID=permission.ID AND
(permission.login ='gast'  OR permission.login ='$user->login')
```

Um Artikel in die Datenbank einzufügen oder bestehende zu verändern oder zu löschen, sind weit weniger anspruchsvolle SQL-Statements vonnöten. Man muss allerdings dafür sorgen, dass jeweils die Tabelle mit den Zugriffsrechten aktuell gehalten wird.

3.6.12.2 DELETE

Beim **Löschen eines Artikels** entsteht das Problem, dass seine Kind-Artikel keinen Vater mehr haben und demzufolge im Navigationsmenü nicht mehr erfasst werden. Man hat hier dasselbe Problem wie beim Löschen eines Knotens aus einem Baum. Mit SQL lässt sich die Schwierigkeit aber leicht meistern: Man setzt zunächst bei allen Artikeln, deren Vater der zu löschende ist, alle *parent*-Attribute auf den Vater des zu löschenden. Danach erst wird der Artikel selbst gelöscht. Aus *permission* werden schließlich alle Einträge entfernt, die sich auf den gelöschten Artikel beziehen.

3.6.12.3 INSERT

Beim **Einfügen eines neuen Artikels** wird der Primärschlüssel automatisch erhöht und ist daher zunächst unbekannt. Damit ist es nicht möglich, die Zugriffsrechte für den Artikel zu setzen. Glücklicherweise bietet PHP eine Funktion `mysql_insert_id()`, die unmittelbar nach dem Einfügen diesen Autoincrement-Wert verrät und so das Problem löst.

Der Name des Autors braucht nicht in einem Formular angegeben zu werden. Da man nur als angemeldeter Benutzer schreiben kann, kann der aktuelle Benutzername automatisch in das SQL-Statement eingebunden werden. Auch in der Tabelle *permission* kann er automatisch eingetragen werden.

3.6.12.4 UPDATE

Das **Update eines Artikels** erfordert nicht nur, dass die Attribute von *artikel* aktualisiert werden, sondern auch die Zugriffsrechte müssen neu gesetzt werden. Am einfachsten löscht man sie zuerst vollständig, um sie dann neu zu setzen. Eine Abfrage, die nur Änderungen berücksichtigt, wäre zu umständlich.

Um die Zugriffsrechte neu setzen zu können, müssen **alle** Benutzer aufgelistet werden, nicht nur die, die bereits Leserecht haben. Wünschenswert ist, dass die letzteren in der Liste bereits selektiert erscheinen.

Mit SQL alleine ist dies nicht machbar. Man muss sich helfen, indem man zunächst die Liste der Leseberechtigten ermittelt und danach, beim Anlegen der Liste aller Benutzer jeweils überprüft, ob

der jeweilige *login* in der Liste der Leseberechtigten erscheint. In diesem Fall kann der Name vorselektiert werden.

Auf jeden Fall muss gewährleistet sein, dass der Autor immer leseberechtigt bleibt, er sich also auch nicht selbst aussperren kann.

In der Beispielimplementierung kann man das Verfahren in `include/edit.php` studieren.

3.6.13 Die Beispielimplementierung

Auf dem beiliegenden Datenträger findet sich eine kommentierte Beispielimplementierung `myco`.

Um sie in Funktion zu sehen, wird zunächst das Verzeichnis `myco` mit allen Unterverzeichnissen an einer beliebigen Stelle auf dem Webserver abgelegt.

Im zweiten Schritt muss die Datenbank angelegt werden. Dazu benutzt man am besten das oben erwähnte phpMyAdmin. Der Name der Datenbank spielt keine Rolle, er kann beispielsweise *schueler1* sein. Für diese Datenbank führt man das beiliegende SQL-Skript (`myco/cms.sql`) aus. Es erzeugt die Tabellen *artikel*, *permission*, *layout* und *user* gemäß dem oben angegebenen ER-Modell.

In `myco/include/dbconnect.php` müssen die Zugangsdaten zur Datenbank eingetragen werden, also der Name des Rechners, auf dem MySQL installiert ist, der Name der Datenbank, der Loginname des Benutzers, unter dessen Identität man auf die Datenbank zugreift und dessen Passwort.

In der Tabelle *layout* müssen einige Pfade gesetzt werden:

URI: Uniform Resource Identifier der Webpräsenz, z.B:
`http://server.de/verzeichnis/myco`

docroot: absoluter Pfad zu `myco` auf dem Server, z.B. `/srv/www/htdocs/myco`

serverroot: relative Verzeichnisangabe auf dem Server, z.B. `/~user/myco`

Hat man `cms.sql` ausgeführt, sind bereits mehrere Benutzer eingerichtet:

login	***Name***	***Vorname***	***Email***	***Passwort***	***Autor***	***Redakteur***	***Administrator***
admin	Istrator	Admin	Ad@localhost	asdf	Y	Y	Y
autor	Tor	Au	autor@localhost	asdf	Y	N	N
autor1	Tor1	Au	autor1@localhost	asdf	Y	N	N
editor	Augstein	Rudolf	ra@localhost	asdf	N	Y	N
Egon	Nichts	Darf	egon@logcalhost	asdf	N	N	N
gast	Besucher	Anonymer	nobody@localhost		N	N	N

Einschränkungen:

Wenn ein Artikel neu angelegt oder bearbeitet wird, ist es möglich, einen untergeordneten Artikel als Vater auszuwählen. Danach wird keiner der Artikel, die sich in der so entstandenen Zirkelbeziehung befinden, angezeigt. Eine Routine, die dies verhindert, wurde aus Zeitgründen nicht geschrieben.

Der so genannte Teaser, der zum Artikelinhalt hinführen soll, ist zwar im ER-Modell enthalten, wird aber noch nicht verwendet. Dasselbe gilt für den Veröffentlichungszeitraum.

3.6.14 Verwendung der Beispielimplementierung im Unterricht.

Ein CMS unter Verwendung von PHP und einer Datenbank zu programmieren, ist ein zeitaufwendiges Unterfangen. Man wird im Unterricht nicht immer die Zeit finden, ein solches Projekt vollständig durchzuführen.

Auf der Basis der Beispielimplementierung `myco` lassen sich aber Aufgabenstellungen finden, die bei Bedarf in einem überschaubaren Zeitrahmen zu lösen sind.

denkbar wäre z.B.:

- Entwicklung eines Autorenmoduls mit einer anderen Programmiersprache. Es existiert ein ODBC-Treiber für MySQL, der es erlaubt, aus Windows-Programmen auf eine MySQL-Datenbank im Netzwerk zuzugreifen. Mit Java oder Delphi könnte man ein Programm realisieren, das das Autorenmodul (edit.php) ersetzt. StarOffice[41] von der Firma Sun in der Version 7 unterstützt direkt MySQL und kann ebenfalls für die Erstellung eines Autorenmoduls verwendet werden.
- Rekonstruktion von gelöschten Funktionen
  Die Schüler erhalten den Quelltext, in dem Funktionen wie das Erzeugen des Navigationsmenüs oder das Darstellen der Artikel nur als Funktionsrumpf enthalten sind. Sie programmieren ein eigene Lösung.
- Erstellen von Masken für die Zeit/Datum-Felder, die im ER-Modell bereits erscheinen, und Modifikation der SQL-Abfragen, so dass der Veröffentlichungszeitraum berücksichtigt wird.
- Entwicklung von neuen Funktionen. Hier bietet sich eine Vielzahl von Möglichkeiten:

Eine so genannte **Sitemap** ist relativ einfach zu erstellen, indem man die Funktion für das Erstellen des Navigationsmenüs als Ausgangspunkt verwendet.

Eine **Suchfunktion** ist durch eine SQL-Abfrage über alle Artikel ebenfalls leicht zu realisieren. Reizvoll wäre eine Maske für eine *query by example*-Abfrage.

Ein **Administrationsmodul** muss lediglich ein Formular für das Bearbeiten der Benutzerliste und des Layout bereitstellen. Da hierfür keine komplexen Zugriffsrechte beachtet werden müssen, sind die nötigen SQL-Befehle einfach. Lediglich beim Löschen von Autoren sollte berücksichtigt werden, dass unter Artikeln deren Name nicht mehr angezeigt werden kann.

Für folgende Erweiterungen muss das ER-Modell modifiziert werden.

Die **Anordnung der Einträge im Navigationsmenü** erfolgt zufällig. Ein Attribut *sortorder* müsste in *artikel* eingefügt werden, um die Reihenfolge beim Editieren festlegen zu können.

Um ein **individuelles Layout** für jeden Artikel oder ein **benutzerspezifisches Layout** zu ermöglichen, müsste jeweils eine neue Beziehung im ER-Modell geschaffen werden.

Um **mehrere Artikel unter einem Menüpunkt** zusammenzufassen, müsste ebenfalls das ER-Modell angepasst werden: durch ein Attribut *menu* könnte festgelegt werden, ob ein Artikel im Menü erscheint oder ob er angezeigt wird, wenn sein *parent* gewählt wurde.

Eine Möglichkeit, **Artikel durch registrierte Benutzer kommentieren** zu lassen, erfordert eine neue Entity *kommentar* und eine Beziehung zwischen *artikel* und *kommentar*.

41 http://wwws.sun.com/software/star/staroffice/index.html

Verzeichniss der Beispiele auf dem Datenträger:

frames1	Beispiel für Frameverwendung
frames2	Frames mit Submenü
cgi-bin	cgi-Verzeichnis
SSI-Demo	Demonstration der Möglichkeiten von Server Side Includes
SSI-exec-cmd	CMS mit SSI, (#exec cmd...)
SSI-include-virtual	CMS mit SSI, (#include virtual...)
SSI-include-virtual2	CMS mit SSI, (#include virtual...), Anzeige von Dateien und Verzeichnissen
php0	Übertragung des SSI-Beispiels nach PHP4
php1	Basis-System mit PHP ohne Frameset
php2	wie php1, Reduzierung auf eine Datei index.php, (QUERY_STRING)
php3	wie php2, mit einfachem Editor (HTML-Formular und Textarea)
php4	wie php3, mit Authentisierung mittels .htaccess
php5	wie php3, mit Authentisierung durch Sessionmanagement
myco	Datenbankbastiertes CMS
dbbrowser	Datenbankbrowser in PHP
sql-editor	php-Skript zum Experimentieren mit SQL

4. Quellenverzeichnis

Die hier angegebenen Hyperlinks waren im September 2003 erreichbar.

Lehrplan Informatik

http://www.hessisches-kultusministerium.de/downloads/lehrpl/gymnasium/Informatik.pdf
Lehrplan Informatik des Landes Hessen

Zur Geschichte des WWW

http://public.web.cern.ch/public/about/achievements/www/history/history.html
Zur Geschichte des WWW bei dem CERN

http://www.w3.org/History/19921103-hypertext/hypertext/DataSources/WWW/Servers.html
Liste der 1992 bekannten WWW-Server

http://news.netcraft.com/archives/web_server_survey.html
Webserverstatistik

http://www.mit.edu/people/mkgray/net/
Statistik zum Wachstum des Internets

http://www.w3.org/History
Archiv des W3-Consortiums zur Geschichte des WWW

http://www.w3.org/History/1991-WWW-NeXT
Der Browser-Editor von Tim Berners-Lee (Dokumentation)

http://www.w3.org/MarkUp/tims_editor
Bildschirmfoto des Browser-Editors von Tim Berners-Lee

http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Provider/Overview.html
Möglichkeiten, Inhalte über HTTP zu publizieren (Tim Berners-Lee)

http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html
Übersicht über die Inhalte des WWW 1992

Software

WWW-Server, HTTP- Protokoll, CGI-Standard

http://www.ietf.org/rfc/rfc2616.txt
Spezifikation des HTTP-Protokolls

http://hoohoo.ncsa.uiuc.edu/
Homepage des NCSA-Webservers, der mittlerweile von Apache abgelöst wurde.

http://hoohoo.ncsa.uiuc.edu/docs/cgi/overview.html
CGI-Dokumentation

http://www.apache.org/
Webserver Apache

http://httpd.apache.org/docs/
Dokumentation zum Webserver Apache

http://httpd.apache.org/docs/howto/htaccess.html
Dokumentation zum Gebrauch von .htaccess-Files

http://httpd.apache.org/docs/mod/mod_include.html#includevirtual

Dokumentation zum Apache-Modul ServerSideIncludes

http://httpd.apache.org/docs/suexec.html

Dokumentation zum suexec-Mechanismus des Apache

http://www.w3.org/Jigsaw/
Der HTTP-Referenzserver des W3-Consortiums

http://www.w3.org/Jigsaw/Doc/User/SSI.html
Dokumentation der ServerSideInclude-Fähigkeiten des JIGSAW-Servers

PHP

http://www.php.net
Die Skriptsprache PHP

http://de.php.net/manual/de/index.php
Deutschsprachige Dokumentation zu PHP

MySQL

http://www.mysql.com/
Die Opensource-Datenbank MySQL

http://www.phpmyadmin.net/
Ein Web-Frontend zu MySQL

http://mysqlfront.venturemedia.de/
Windows-Frontend zu MySQL

http://wwws.sun.com/software/star/staroffice/index.html
Office-Programm von Sun mit integriertem MySQL-Treiber

Linux

http://www.suse.de
Eine gängige Linux-Distribution

Sonstige

http://www.contentmanager.de/itguide/marktuebersicht.html
Marktübersicht von Contentmanagementsystemem

http://c2.com/cgi/wiki
Editierbare Websites

Ich versichere, diese Arbeit selbständig verfasst, keine anderen als die angegebenen Hilfsmittel verwendet und die Stellen, die in anderen Werken im Wortlaut, als Graphik oder dem Sinne nach entnommen sind, mit Quellenangaben kenntlich gemacht zu haben.