Berufsakademie Sachsen
Staatliche Studienakademie Leipzig

Einführung eines Mehrbenutzersystems für die Administration eines Web-Portals

Praxisarbeit in der Fachrichtung Informatik
im 3. Semester

Eingereicht von: Christian Gräfe
Demmeringstraße 8b
04177 Leipzig
Seminargruppe: IT05
Matrikelnummer: 205830

Betreuer: Prof. Dr. habil. Harald Englisch
Beratung Medizin-Informatik
Burghausener Str. 18
04178 Leipzig, OT Böhlitz-Ehrenberg

Leipzig, 10.01.2007

Inhaltsverzeichnis

1 Einführung / Problemstellung

Übersteigt der Umfang eines Projektes eine gewisse Grenze, ist es beinahe unmöglich als Einzelperson die Qualität des Projektes konstant zu halten. Um die Qualität zu erhalten bzw. zu steigern, sind weitere Arbeitskräfte von Nöten. Aus diesem Zusammenhang ergeben sich neue Anforderungen an das Projektmanagement. Es wird nun eine Aufgaben- bzw. Zuständigkeitsverteilung benötigt. Es muss geklärt werden, wer welche Änderungen durchführen darf. Da die Arbeiten nun nicht mehr durch eine Einzelperson erfolgen, muss die Dokumentation der getätigten Arbeiten an dem Projekt ebenfalls neu strukturiert werden. Es muss nun nachvollziehbar sein, wer wann eine Änderung vorgenommen hat.

Überträgt man das Ganze nun auf ein Onlineprojekt, kommen zu den schon genannten Punkten weitere hinzu. Bei Onlineprojekten muss ein besonderer Wert auf die Absicherung vor fremden Zugriff gelegt werden. Es dürfen nur Änderungen durch autorisierte Personen durchgeführt werden. Diese Änderungen müssen von überall her ausgeführt werden können, also nicht nur von einem bestimmten PC aus, sondern zum Beispiel auch von dem Privat-PC zu Hause oder wenn man gerade bei einem Kunden ist, vom Kunden-PC aus.

In dieser Praxisarbeit geht es um ein Onlineprojekt, wo nun genau dieser Fall eingetreten ist. Das Projekt ist zu groß geworden, um es allein zu bearbeiten. Um das Projekt weiter so qualitativ fortführen zu können wie bisher, werden die Aufgaben auf mehrere Personen aufgeteilt. Dabei hat allerdings nicht jeder Benutzer die gleichen Rechte, manche Punkte dürfen nur durch ausgewählte Nutzer bearbeitet werden.

2 Vorbetrachtungen

2.1 Auswertung der Gegebenheiten

Bei dem hier zu betrachtenden Online-Projekt handelt es sich um das Gesundheits-Portal „Gesundheit-Sachsen“. Das Portal bietet Adressen, Nachrichten, Termine, Bücher und Links zum Thema Gesundheit. Die Daten sind in einer Datenbank erfasst und werden dynamisch auf der Internetseite angezeigt. Des Weiteren wird einmal wöchentlich ein Newsletter an eingetragene User versendet. In der Datenbank befinden sich zusätzlich zu den schon genanten Daten noch Bannerdaten, Daten für Werbeanzeigen im Newsletter und statistische Daten, wie zum Beispiel Logs für die Bannerklicks oder Seitenaufrufe.

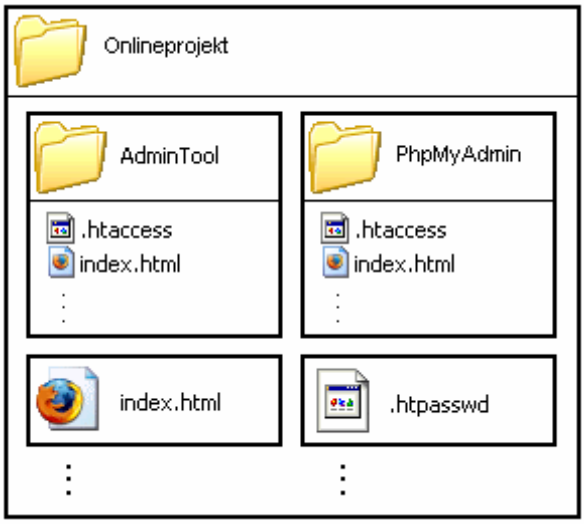

Abbildung 1: Ordnerstruktur

Zurzeit werden die Daten mit Hilfe eines eigenen Administrationstools und mit dem Programm PhpMyAdmin in die Datenbank eingepflegt. Die zwei Programme liegen jeweils in einem eigenen Ordner, der durch eine .htaccess-Datei per Passwort geschützt ist. Zum Arbeiten mit den Programmen gibt es einen Login, der für beide gleich ist. Es existiert eine gemeinsame .htpasswd-Datei, in der ein Benutzer eingetragen ist. (siehe Abb. 1)

Wie schon durch die Benutzung der Dateien .htaccess und .htpasswd zu vermuten war, liegt die Internetseite auf einem Apache-Server. Zur dynamischen Darstellung der Webseiteninhalte werden PHP-Skripte genutzt, die die Daten aus einer MySql-Datenbank abrufen und für die Anzeige aufbereiten. Das ganze wird unter dem Betriebssystem Linux ausgeführt, somit handelt es sich hierbei um ein LAMP-System (LAMP ist das Akronym für Linux Apache MySql PHP).

Es wäre nun möglich, einfach mehrere Benutzer in die .htpasswd-Datei zu schreiben, allerdings hat dann jeder Benutzer die gleichen Rechte. Da die Rechte ebenfalls beschränkt werden sollen, genügt diese Erweiterung nicht den Ansprüchen. Des Weiteren ist unter Verwendung der .htaccess kein Logout möglich, da dies in den Apache-Spezifikationen nicht vorgesehen ist.

Das Admintool setzt sich aus mehreren PHP-Skripten zusammen, mit denen die Daten aus der Datenbank verändert werden können bzw. neue Daten in die Datenbank eingefügt werden können. Die Skripte sind nach ihrem Teilgebiet in Ordner verteilt, zum Beispiel ein Ordner „News“ mit allen Skripten für die Bearbeitung von Nachrichten (siehe Abb. 2). Die Steuerung erfolgt über eine index.html-Datei, die zwei Frames enthält. Im linken Frame befindet sich die Navigation mit Hyperlinks zu den verschiedenen Skripten. Diese Skripte werden nach einem Klick auf den entsprechenden Link auf der rechten Bildschirmseite angezeigt (siehe Abb. 3).

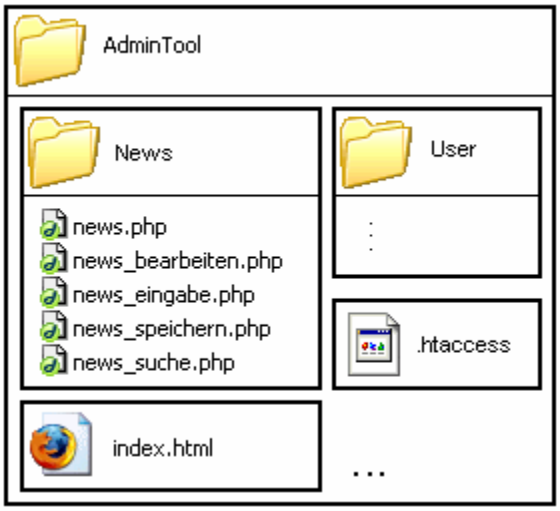

Abbildung 2: Verzeichnisse Admintool

Das derzeitige Admintool wurde vom Autor dieser Praxisarbeit programmiert, da es nun allerdings nicht mehr den Anforderungen entspricht, muss es überarbeitet werden. Um die Produktivität des ganzen Projektes zu steigern, wird das Programm gleichzeitig um einige Komponenten erweitert. Da diese Erweiterungen nichts mit dem Thema dieser Praxisarbeit zu tun haben, werden sie nicht extra erläutert.

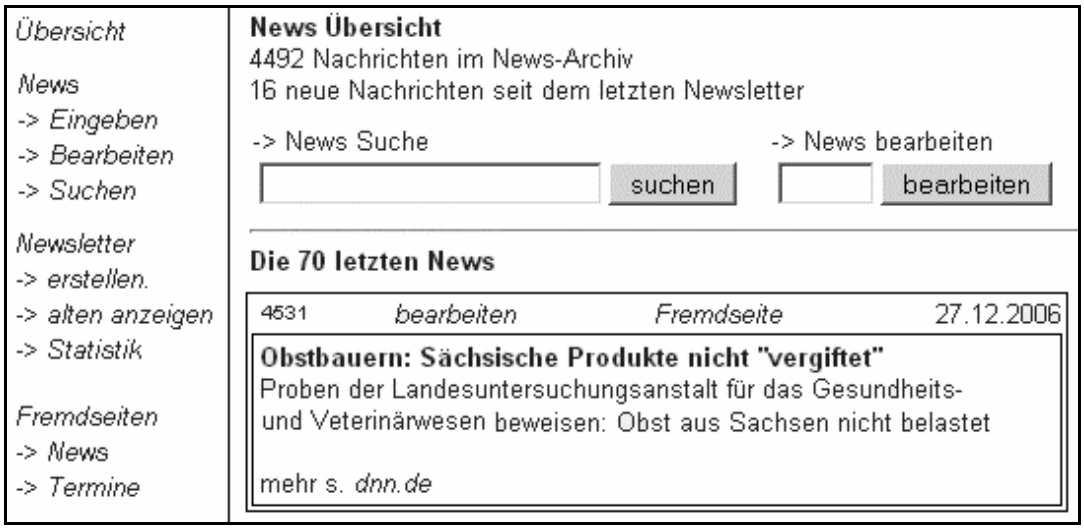

Abbildung 3: Altes Admintool

2.2 Nötige Anpassungen an das System

Wie im vorangegangenen Kapitel bereits erwähnt, reicht es nicht aus, einfach nur mehrere Benutzer in die .htpasswd einzutragen. Zum einen lässt sich nicht sicherstellen, dass der Benutzer nur Zugriff auf bestimmte Teile des Admintools hat und zum anderen ist kein Logout möglich. Somit muss ein anderer Lösungsweg gewählt werden. Da die Internetseite auf einem LAMP-System läuft, bietet es sich an, eine Lösung über PHP und MySql zu realisieren. Die Benutzer können in der Datenbank gespeichert werden und über ein Loginskript erhalten sie Zugang zum Admintool.

Als kleines Problem erweist sich bei dieser Lösung allerdings die Einbindung des Programms PhpMyAdmin, da die Benutzung dieses Programms nur über eine .htaccess gesperrt werden kann. Da nur sehr wenige Benutzer später dieses Programm verwenden dürfen, kann dieses Problem erst einmal vernachlässigt werden und der Zugriff weiterhin durch die .htaccess gesteuert werden. In der zugehörigen .htpasswd-Datei werden später automatisiert die autorisierten Benutzer stehen.

Damit der Zugriff zuverlässig gesteuert werden kann, muss der grundsätzliche Aufbau des Admintools verändert werden. Es ist wesentlich einfacher und sicherer, wenn die Einzelskripte aus einem zentralen Skript heraus aufgerufen werden und nicht selber angesprochen werden können. Bei dieser zentralen Steuerung kann der Zugriff auf bestimmte Funktionen/Skripte durch Rechte beschränkt werden. Die Skripte mit den Funktionen müssen allerdings so angepasst werden, dass sie für sich selber nicht lauffähig sind. Dieses Verhalten lässt sich erreichen, indem man im zentralen Skript eine Variable definiert, die in den Einzelskripten abgefragt wird. Eine weitere Möglichkeit ergibt sich durch die Arbeit mit der Datenbank. Wenn die Verbindung zur Datenbank nur im zentralen Skript geschieht, können die Einzellskripte nicht mit der Datenbank arbeiten. Dieser Lösungsansatz wurde bei diesem Projekt verwendet.

Das zentrale PHP-Skript „index.php“ zeigt beim Start, wenn man noch nicht eingeloggt ist, automatisch nur eine Loginseite. Nach dem erfolgreichen Einloggen kann man mit dem Programm arbeiten. Der prinzipielle Aufbau des Programms bleibt aus Übersichtlichkeit erhalten, links die Navigation und rechts die Informationen. Da die angezeigte Seite nun nur noch aus einer Datei besteht, werden die Frames durch eine zweispaltige Tabelle ersetzt, wobei die Inhalte auf der rechten Seite mittels include(), in Abhängigkeit von einer SeitenID, eingebunden werden.

3 Sicherheitskonzept

3.1 Das Login

Das Login des Admintools ist die wichtigste Sicherheitsvorrichtung des Systems, daher muss es besonders gut abgesichert sein. Das Loginskript zeigt zwei Eingabefelder an, eines für den Benutzernamen und eines für das Passwort. Während einer Eingabe in das Passwortfeld werden, wie es in so einem Feld üblich ist, nur Sterne angezeigt. Nach einem Scheitern des Loginversuches wird nicht angegeben, ob der Benutzername oder das Passwort falsch waren (siehe Abb. 4). Somit wird verhindert, dass man durch bloßes Probieren einen Benutzernamen findet und bei diesem dann nach dem Passwort sucht.

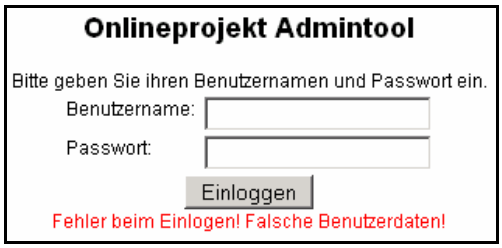

Abbildung 4: Fehler beim Login

Die Übergabe der eingegebenen Daten erfolgt mittels POST, da bei der Übergabe über die URL (mittels GET) das Passwort unverschlüsselt angezeigt wird und dies ein viel zu hohes Sicherheitsrisiko darstellt.

Ein sehr wichtiger Aspekt bei den Sicherheitsvorkehrungen heutzutage ist die Absicherung gegen SQL-Injektions.[1] Mit einfachen Mitteln lässt sich so manche Passwortabfrage aushebeln. Wenn man als SQL-Anweisung zur Passwortkontrolle zum Beispiel den folgenden String verwendet:

```
SELECT id FROM user WHERE name='$loginname' AND pass='$loginpasswort'
```

Man braucht nur für loginname und loginpasswort den String *' OR ''='* verwenden und wird erfolgreich authentifiziert. Wenn man sich die vollständige Abfrage ansieht, wird der Grund deutlich:

```
SELECT id FROM user WHERE name='' OR ''='' AND pass='' OR ''=''
```

Wie man auf das Ergebnis dieser Abfrage reagiert, kann zur Sicherheit des ganzen Loginsystems beitragen. Diese SQL-Anweisung liefert in der Regel mehr als ein Ergebnis zurück. Es liefert die komplette Spalte „id“ der Tabelle „user“ als Ergebnis, da im WHERE-Statement zusammengefasst „True“ steht. Da wir allerdings wissen, dass diese Anweisung nur einen Benutzer zurückliefern darf, sollte dies bei dem Ergebnis der SQL-Abfrage mit überprüft werden. Man kann sich in PHP die Anzahl der Reihen einer MySql-Abfrage mithilfe des Befehl mysql_num_rows() anzeigen lassen.

[1] Vgl. http://www.inspire-world.de/board/showthread.php?t=14060

In einem PHP-Skript sieht dies folgendermaßen aus:[2]

```
$erg=mysql_query("SELECT id FROM user WHERE name='$loginname'
                  AND pass='$loginpasswort'");
if (mysql_num_rows($erg)!=1) echo "FEHLER!";
else …
```

Damit solche Eingaben wirkungslos sind, müssen die übergebenen Parameter „escaped“ werden. Das bedeutet, dass Hochkommas und andere Sonderzeichen mit einem Schrägstrich versehen werden und dadurch in der SQL-Abfrage als Text erkannt werden und nicht als Teil der Abfrage. PHP stellt dafür die Funktion mysql_real_escape_string() zur Verfügung.[3]

[2] http://de2.php.net/mysql_num_rows

[3] http://de2.php.net/mysql_real_escape_string

3.2 Speicherung des Passwortes

Ein weiterer Punkt, um die Sicherheit der Benutzerdaten zu gewährleisten, besteht darin, die Passwörter nicht im Klartext in der Datenbank zu speichern, sondern verschlüsselt. Dabei muss allerdings darauf geachtet werden, dass die gespeicherten Daten nicht entschlüsselt werden können, da dies ein Sicherheitsrisiko darstellt. Wenn ein Unbefugter die Daten aus der Benutzerdatenbank erhält und die Passwörter entschlüsselt, kann er sich als irgendein Nutzer einloggen und Schaden anrichten.

So genante Ein-Wege-Verschlüsselungen sind hier die Lösung. Die bekanntesten Verschlüsselungen sind DES, MD5, SHA1 und Blowfish. Da als DBMS MySql verwendet wird, können dessen interne Verschlüsselungsmethoden genutzt werden. Da später das gespeicherte Passwort auch für den Login über .htaccess genutzt werden soll, bietet sich die Verschlüsselung DES an. MySql stellt dafür die Funktion ENCRYPT() zur Verfügung.[4] Mit dieser Funktion erzeugt MySql in Abhängigkeit eines „salt“, sprichwörtlich das Salz in der Suppe, einen Hashcode des Passwortes. Aus diesem so erzeugten Hashcode kann das ursprüngliche Passwort nicht wiederhergestellt werden. Um nun das Passwort beim Login überprüfen zu können, muss der eingegebene Wert mit dem gleichen „salt“ verschlüsselt und mit dem gespeicherten Hashcode aus der Datenbank verglichen werden.

```
SELECT id FROM user WHERE name='$loginname'
AND pass=ENCRYPT('$loginpasswort','at')
```

Stimmen die Werte nicht überein, liefert diese Abfrage ein leeres Resultat zurück. War die Abfrage allerdings erfolgreich, kann gleich die UserID des Benutzers gespeichert werden, damit der Benutzer mit dem Admintool arbeiten kann.

[4] http://dev.mysql.com/doc/refman/5.1/de/encryption-functions.html

3.3 Absicherung der Rechte

Eine der ersten Aktivitäten der index.php ist das Abfragen, ob eine UserID gespeichert ist. Wenn dies nicht der Fall ist, erscheint die Loginaufforderung. Damit die UserID nicht eingeschleust werden kann, zum Beispiel über die URL mittels Parameter (http://www.onlineprojekt.de/admin/index.php?userid=1 o.ä.), wird nach der UserID explizit in den Sessiondaten gesucht, also im Array $_SESSION.

Mithilfe dieser UserID holt sich das Skript die Rechte des Benutzers und wertet diese aus (siehe nachfolgendes Kapitel). Damit der Nutzer nur das machen kann, wofür er auch Rechte besitzt, gibt es mehrere Absicherungen.

a) Der Nutzer sieht nur das, was er auch benutzen darf

In der Navigation wird nur auf Skripte bzw. Seiten verlinkt, die der Benutzer wirklich betreten darf. Wenn der Nutzer zum Beispiel das Recht „News“ hat, wird in der Navigation eine Box mit allen Links für die Nachrichten angezeigt (eingeben, bearbeiten, suchen, …).

b) Die Skripte werden in Abhängigkeit der Rechte eingebunden

Wie bereits in Kapitel 2.2 angegeben, werden die Funktionen des Admintools über eine SeitenID angesprochen. In Abhängigkeit von dieser SeitenID werden unterschiedliche Skripte für die Anzeige auf der rechten Bildschirmseite eingebunden. Die Auswertung der SeitenID erfolgt mittels switch(). In jedem case-Zweig wird nun überprüft, ob der Nutzer das Recht hat diese Seite anzuzeigen. Ist dies der Fall, wird das entsprechende Skript eingebunden, andernfalls sieht der Benutzer eine leere Seite.

```
$ur=get_userrechte($_SESSION['userid']);
switch($seitenid)
{
case 100: if($ur['news']) include('include/news/news.php'); break;
case 101: if($ur['news']) include('include/news/news_bearbeiten.php'); break;
…
case 200: if($ur['termine']) include('include/termin/termine.php'); break;
case 201: if($ur['termine']) include('include/termin/termin_bearbeiten.php'); break;
…
}
```

c) Die Einzelskripte sind alleine nicht lauffähig

Da in der index.php die Verbindung zur Datenbank hergestellt wird, sind die Einzelskripte für sich alleine nicht lauffähig, erst durch das Einbinden über include() funktionieren sie.

4 Benutzer-/Rechteverwaltung

Im vorangegangenen Kapitel wurden bereits die UserID, der Benutzername und das Benutzerpasswort erwähnt. Dies sind nicht die einzigen Daten die vom Benutzer in der Datenbank gespeichert werden müssen. Um mit dem Benutzer persönlich kommunizieren zu können, werden in der Datenbank noch der vollständige Name, die Mailadresse und zusätzliche Bemerkungen gespeichert.

Aus den Anforderungen ergibt sich, dass die Benutzer nicht alle die gleichen Rechte haben dürfen. Es gibt mehrere Möglichkeiten die Rechte zu speichern. Zum einen könnte man die Rechte getrennt von den Benutzerdaten speichern, also in einer eigenen Tabelle. In dieser Tabelle erfolgt dann ein Eintrag mit der BenutzerID und dem Recht, das der Nutzer erhält, also zum Beispiel das Recht „News bearbeiten“. Der Nutzer hat nur die Rechte, die er in der Tabelle „Rechte“ zugeteilt bekommen hat. Bei dieser Art der Speicherung ist es ohne Probleme möglich, neue Rechtearten hinzuzufügen. Sie bietet sich vor allem an, wenn man viele Benutzer hat oder unterschiedliche Benutzergruppen unterstützt will.

Bei wenigen Benutzern genügt es, die Rechte direkt mit in die Tabelle zu speichern, in der auch die Benutzerdaten stehen. Da die Rechte bei dieser Variante direkter Bestandteil der Tabellenstruktur sind, ist es nicht ohne weiteres möglich neue Rechte einzuführen. Die Speicherung erfolgt dann in der Form, dass der Spaltenname die Bezeichnung des Rechtes ist und in der Spalte eine 0 oder eine 1 gespeichert wird. Diese Variante wurde bei diesem Onlineprojekt gewählt, da es nur wenige Benutzer gibt. Zur Speicherung der Rechte wurde in den Spalten der kleinste MySql-Datentyp TinyInt (1Byte) gewählt.[5] Links ist eine SQL-Abfrage dargestellt, die solch eine Tabelle erzeugt.

```
CREATE TABLE `adminuser` (
  `id` tinyint(2) NOT NULL auto_increment,
  `name` varchar(50) NOT NULL default '',
  `pass` varchar(32) NOT NULL default '',
  `real_name` varchar(255) NOT NULL default '',
  `news` tinyint(1) NOT NULL default '0',
  `termine` tinyint(1) NOT NULL default '0',
  `books` tinyint(1) NOT NULL default '0',
  `statistik` tinyint(1) NOT NULL default '0',
  `banner` tinyint(1) NOT NULL default '0',
  `anzeigen` tinyint(1) NOT NULL default '0',
  `bem` varchar(255) NOT NULL default '',
  PRIMARY KEY  (`id`)
);
```

[5] http://dev.mysql.com/doc/refman/5.1/de/numeric-type-overview.html

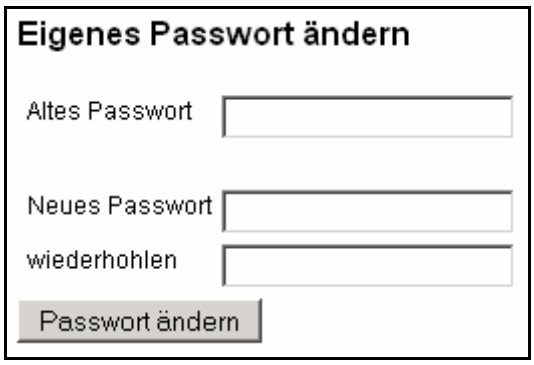

Abbildung 5: Passwort ändern

Damit nun die Benutzerdaten und die Rechte auf einfache Art geändert werden können, wird in das Admintool eine Erweiterung für die Benutzerverwaltung eingebaut. Diese Benutzerverwaltung ist ebenfalls über ein eigenes Recht geschützt, damit nicht jeder Benutzer damit arbeiten kann. Das einzige, was jeder Nutzer ändern kann, ist sein eigenes Passwort. Um dieses ändern zu können, muss vorher allerdings das alte Passwort eingegeben werden (siehe Abb. 5). Falls der Benutzer vergessen hat sich auszuloggen, kann ihm somit nicht durch einen Fremdnutzer das Passwort verstellt werden.

Da für die Arbeit mit dem Admintool die UserID in den Sessiondaten erforderlich ist, lässt sich ganz einfach eine Option zum Ausloggen einfügen. Mithilfe der PHP-Funktion session_destroy() wird, wie der Name schon vermuten lässt, die aktuelle Session zerstört und die Sessiondaten gelöscht. Da nun keine UserID mehr angegeben ist, erscheint wieder die Loginseite.

Sollte der Nutzer sein Passwort vergessen haben, gibt es über die Benutzerverwaltung des Admintools die Möglichkeit ein neues Passwort zu setzen. In diese Benutzerverwaltung dürfen, wie schon erwähnt, nur Nutzer, die das entsprechende Recht haben. Damit man bei etwaigen Fehlern immer noch mit dem Admintool arbeiten kann, gibt es einen Superuser, der mit allen Rechten ausgestattet ist.

5 Protokollierung von Aktivitäten

Da jetzt mehrere Benutzer Daten eintragen und ändern können, soll eine Protokollierung in das System eingebunden werden. Weil die Skripte zum Speichern von Änderungen oder Neueintragungen unabhängig von den Bearbeitungsskripten sind, genügt es diese Speicherskripte zu überarbeiten. Hier muss nun nach Erfolg der eigentlichen Speicheranweisung protokolliert werden, was, wann, wo, von wem gespeichert wurde.

Es bietet sich an, dieses Log in eine Tabelle der Datenbank zu speichern, da das Log viele Fremdschlüssel zu anderen Tabellen enthalten wird. Die eigentlichen Daten, wie zum Beispiel Termine, Bücher oder Banner, sind in jeweils einer eigenen Tabelle gespeichert, in denen als Primärschlüssel eine ID fungiert. Als Primärschlüssel für die Logtabelle bieten sich nun eine Tabellenummer und die Nummer des Elementes an. Da auf einfache Art und Weise gespeichert werden soll, was mit dem Objekt gemacht wurde, bearbeitet oder neu eingetragen, benötigt man noch eine dritte Spalte für den Primärschlüssel. Es genügt, wenn man hier eine 0 (neu eingetragen) oder eine 1 (bearbeitet) speichert.

Unter diesem dreiteiligen Primärschlüssel werden nun der Benutzer, also die aktuelle UserID und der aktuelle Zeitstempel gespeichert. Die SQL-Anweisung zum Erzeugen solch einer Tabelle sieht folgendermaßen aus:

```
CREATE TABLE `admin_log` (
  `tid` tinyint(3) unsigned NOT NULL default '0',
  `eid` smallint(5) unsigned NOT NULL default '0',
  `sid` tinyint(3) unsigned NOT NULL default '1',
  `userid` tinyint(3) unsigned NOT NULL default '0',
  `time` timestamp NOT NULL,
  PRIMARY KEY  (`tid`,`eid`,`sid`)
);
```

Bei dem Einfügen eines Logeintrages wird REPLACE genutzt, da hierbei entweder ein neuer Datensatz angelegt oder ein vorhandener überschrieben wird.[6] Die Anweisung um solch einen Eintrag vorzunehmen, ist in allen Speicherskripten gleich:

```
mysql_query("REPLACE INTO admin_log VALUES
                ($tid,$eid,".(($neu)?('0'):('1')).",".$_SESSION['userid'].",NOW())");
```

[6] http://dev.mysql.com/doc/refman/5.1/de/replace.html

6 Einbindung eines Fremdprogramms

Da sich, wie anfangs schon erwähnt, das Programm PhpMyAdmin nicht auf die gleiche Art und Weise in das Admintool einbinden lässt wie die Einzelskripte, wurde hier ein anderer Ansatz gewählt.

In der Benutzertabelle gibt es ein Recht „pma“. Sobald dieses Recht bei einem Benutzer gesetzt ist, hat er im Menü einen Link „PhpMyAdmin“, der auf das Verzeichnis des gleichnamigen Programms zeigt. Dieses Verzeichnis ist nun wiederum mittels .htaccess und .htpasswd geschützt. Der Login für das Verzeichnis, ist der gleiche Login, wie für das Admintool, mit dem einzigen Unterschied, dass man die Daten nun in eine Login-Messagebox des Browsers eingeben muss.

Damit der Login bei dem Admintool und bei PhpMyAdmin wirklich der gleiche ist, müssen die Daten der .htpasswd mit den aktuellen Benutzerdaten aus der Datenbank übereinstimmen. Um dies zu gewährleisten, wird bei jeder Passwort- bzw. Benutzernamenänderung die Datei .htpasswd neu geschrieben. In dieser Datei stehen allerdings nur Benutzer, die das Recht „pma“ haben. Da der Apacheserver auch die DES-Verschlüsselung beherrscht, können die Passwörter aus der Datenbank eins zu eins übernommen werden und mit dem jeweiligen Benutzernamen zusammen in der Datei .htpasswd gespeichert werden. Eine einfache PHP-Funktion, die diese Aufgabe übernimmt, sieht folgendermaßen aus:

```
function write_passwd()
{
  $file=fopen("./.htpasswd","w");  // Datei zum schreiben öffnen
  if($file)  // Konnte die Datei geöffnet werden?
  {
    $erg=mysql_query("SELECT name,pass FROM gs_user WHERE pma=1");
    while($row=mysql_fetch_array($erg))
          fwrite($file,$row['name'].":".$row['pass']."\n");   // Benutzer schreiben
    fclose($file);  // Datei wieder schließen
  }
  else echo "Fehler beim Öffnen der Passwort-Datei!";
}
```

Einen Nachteil hat diese Absicherung über .htaccess allerdings, es ist kein Logout möglich. Der Benutzer sollte also nur an vertrauenswürdigen Rechnern mit dem Programm PhpMyAdmin arbeiten.

7 Abschließende Bemerkungen

Vor Beginn der Umstellung der Administration auf Mehrbenutzerbetrieb stellte sich die Frage, das gesamte Projekt auf Standartsoftware umzustellen. Es gibt Content Managment Systeme (CMS), die solche Mehrbenutzerfähigkeit für die Administration in einem noch viel größerem Umfang umsetzen. Das Projekt selber läuft allerdings schon sehr gut und zuverlässig in der derzeitigen Form. Für die Administration stehen ebenfalls Mittel zur Verfügung. Eine Umstellung des Projektes auf ein Standard CMS hätte keinen wirklichen Mehrnutzen gebracht.

Wie in den vorangegangenen Kapiteln dargestellt, lässt sich eine Mehrbenutzerfähigkeit mit wenigen Änderungen in ein bestehendes System einbinden. Wenn alles durch ein zentrales Skript gesteuert wird, lassen sich hier die wichtigsten Sicherheitsvorkehrungen, wie Loginabfrage und Einhaltung der Rechte, einbinden. Bei der Umsetzung des ganzen muss allerdings auch ein Auge auf die Sicherheit des Projektes gelegt werden. Die Loginseite bietet den größten Angriffspunkt, wenn sich hier Schwächen zeigen, müssen diese umgehend behoben werden. Als zweiter wichtiger Sicherheitsaspekt gilt die Einhaltung der Rechte. Wenn der Nutzer nicht sieht, was er nicht darf, weiß er auch nicht, wo er manipulieren kann. (siehe Abb. 6)

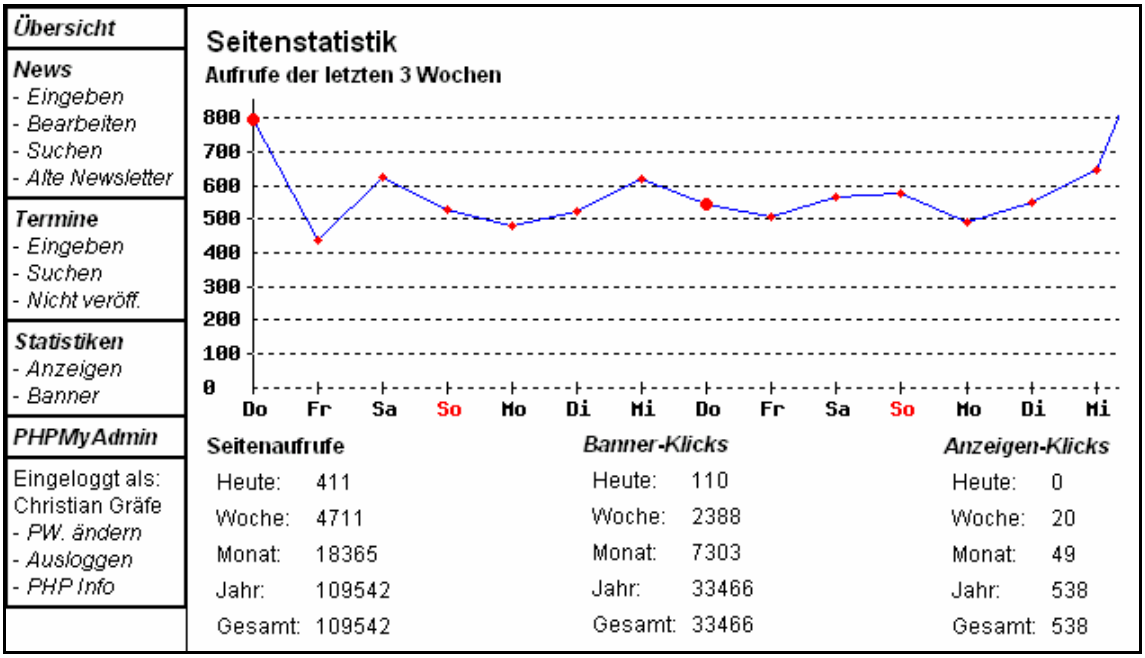

Abbildung 6: Neues AdminTool mit eingeloggtem Autor (mit 4 von 10 Rechten)

Literaturverzeichnis

[1] Leitfaden zur Sicheren PHP-Programmierung (Stand: 13.10.2006)
Internet: http://www.inspire-world.de/board/showthread.php?t=14060

[2] PHP-Dokumentation: mysql_num_rows (08.01.2007)
Internet: http://de2.php.net/mysql_num_rows

[3] PHP-Dokumentation: mysql_real_escape_string (08.01.2007)
Internet: http://de2.php.net/mysql_real_escape_string

[4] MySql: Verschlüsselungs- und Kompressionsfunktionen (08.01.2007)
Internet: http://dev.mysql.com/doc/refman/5.1/de/encryption-functions.html

[5] MySql: Überblick über numerische Datentypen (08.01.2007)
Internet: http://dev.mysql.com/doc/refman/5.1/de/numeric-type-overview.html

[6] MySql: REPLACE (08.01.2007)
Internet: http://dev.mysql.com/doc/refman/5.1/de/replace.html

Abkürzungsverzeichnis

Abkürzung	Bedeutung
CMS	Content Managment System
DBMS	Datenbank Managment System
DES	Data Encryption Standard
LAMP	Linux Apache MySql PHP
MD5	Message Digest Algorithm 5
PMA	PhpMyAdmin
PHP	PHP: Hypertext Preprocessor
SHA1	Secure Hash Algorithm
URL	Uniform Resource Locator

Tabellen- und Abbildungsverzeichnis

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Praxisarbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder veröffentlicht, noch einer anderen Prüfungsbehörde vorgelegt.

Leipzig, 10.01.2007 _____

Christian Gräfe