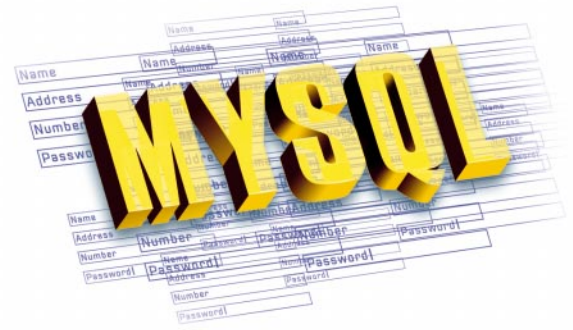


PHP und MySQL

# Eine Mitglieder-Verwaltung



Auf fast allen **interessanten Webseiten** kann man Mitglied werden. Dadurch stehen einem spezielle Funktionen zur Verfügung. Voraussetzung dafür ist jedoch eine **Mitgliederverwaltung**. Mit Hilfe dieses Beitrages implementieren Sie Ihre eigene Mitgliederverwaltung mit **PHP und MySQL**.

THOMAS WÖLFER

Im Rahmen dieses Beitrags implementieren Sie nicht nur die Mitgliederverwaltung, sondern auch ein paar Beispiele für die Nutzung der Verwaltung. Auf der Heft-CD finden Sie dazu vier Dateien: login.php3, user.php, useradmin.php3 und userdb.php3.

Userdb.php3 ist dabei das Back-End der Mitgliederverwaltung. Es enthält eine PHP-Klasse, die den kompletten benötigten Funktionsumfang implementiert. Auf Ihrem Web-Server müssen Sie diese Datei in einem geschützten Bereich unterbringen, auf den Ihre Anwender keinen Zugriff haben. Diese Datei gehört in ein Verzeichnis, das per http nicht zu erreichen ist. In Dateien, in denen Sie die Mitgliederverwaltungsfunktionen benötigen, inkludieren Sie diese Datei per „require“ Statement.

Angenommen der Pfad zu Ihren öffentlichen HTML-Seiten sieht so aus:

```
/usr/home/my_site/htdocs
```

In diesem Fall wäre htdocs das Verzeichnis mit Ihrer „index.html“, die Verzeichnisse darunter sind über http nicht erreichbar. In diesem Fall würde es sich etwa anbieten im Verzeichnis my\_site ein weiteres Verzeichnis mit dem Namen „inc“ anzulegen. In diesem Verzeichnis legen Sie die PHP-Dateien ab, die in einen geschützten Bereich gehören. Die userdb.php3 würden Sie in anderen PHP-Dateien so einbinden:

```
require(„./usr/home/my_site/inc/  
userdb.php3“);
```

Im Beispiel-Code wurde das aus Gründen der Übersichtlichkeit nicht ge-

macht: Hier liegen einfach alle Dateien im gleichen Verzeichnis.

Die Datei userd.php enthält ein Beispiel für die Nutzung der Mitgliederverwaltung. Hier wird ein Formular angezeigt, mit dem Interessierte sich als neue Mitglieder eintragen können. In login.php3 finden Sie ein Beispiel für eine Log-in-Seite.

Mitglieder können sich dort mit einem Formular bei Ihnen anmelden. Klappt die Authentifizierung, werden Sie mit Ihrem Namen, ihrem User-Level und dem Datum der letzten Anmeldung begrüßt.

Useradmin.php3 schließlich enthält die elementare User-Verwaltung. Hier können Sie nicht nur alle Mitglieder auflisten, sondern auch einzelne Mitglieder löschen und die User-Level der Mitglieder setzen. Der User-Level wurde bereits zweimal angesprochen, aber bisher nicht erläutert. Das soll an dieser Stelle nachgeholt werden. Die vorliegende Mitgliederverwaltung ist nur ein Beispiel.

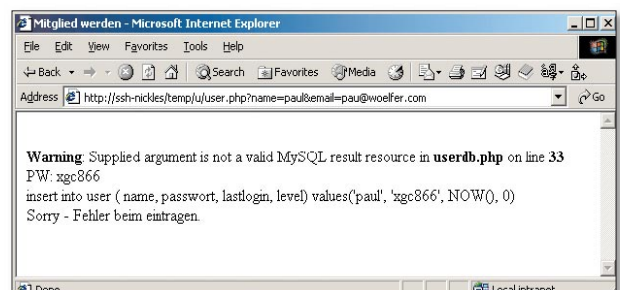
Zwar finden Sie alles, was Sie für eine „echte“ Benutzerverwaltung benötigen, allerdings sind nicht sonderlich viele Daten pro Mitglied vorgesehen. Das können Sie dann ganz nach Bedarf selbst ausbauen. Pro Mitglied werden im vorliegenden Beispiel folgende Daten gespeichert:

- Eine eindeutige ID des Mitgliedes
- Der Benutzername des Mitgliedes, der vom Benutzer freigegeben werden kann
- Das Passwort des Mitgliedes
- Das Datum, an dem sich das Mitglied zum letzten Mal angemeldet hat
- Der User-Level des Mitgliedes

Dabei wird zwischen drei User-Levels unterschieden. Gast, Power-User und Admin. Diese Unterscheidung ist dafür gedacht, dass Sie unterschiedlichen Mitgliedern unterschiedliche Möglichkeiten einräumen können. Ein Gast kann zum Beispiel einfach nur „ordentliches“ Mitglied sein, ein Power-User hätte vielleicht Zugriff auf bestimmte Tools Ihrer Webseite, wie zum Beispiel ein Tool zum Korrigieren Ihrer Link-Liste, während eine Admin Zugriff auf alle Möglichkeiten hat. So könnte vielleicht ein Admin auch Mitglieder aus der Datenbank löschen.

## Das Back-End

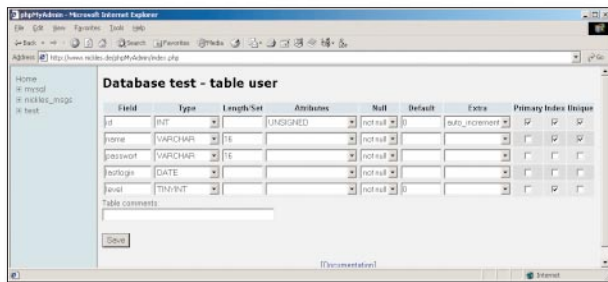
Für die Mitgliederverwaltung benötigen Sie eine Datenbank, in der Sie die Mit-



**WENN DIE ARGUMENTE** an MySQL nicht richtig übergeben werden, erhalten Sie entsprechend merkwürdige Fehlermeldungen.



gliederdaten unterbringen können. Dafür verwenden Sie am besten das Datenbanksystem MySQL, da dieses bei vielen Hosting-Providern von Haus aus mit angeboten wird. MySQL administrieren Sie am besten mit phpMyAdmin.



**MIT PHPMYADMIN** legen Sie die Tabelle für die User-Verwaltung an.

Eine kurze Einführung in dieses Tool finden Sie im Artikel „Gästebuch mit PHP und MySQL“ in diesem Sonderheft. Daher wird hier die Bedienung von phpMyAdmin nicht weiter erläutert.

Um die Daten in MySQL ablegen zu können, benötigen Sie eine Datenbank und eine Tabelle innerhalb dieser Datenbank. Im Beispiel-Code hat die Datenbank den Namen „test“ und die Tabelle trägt den Namen „user“. Diese Parameter (und auch die Zugangs-Passwörter zu MySQL) können sich auf Ihrem System von denen im Beispiel-Code unterscheiden. Um genau zu sein,



**MIT DEM ANMELDEFORMULAR** kann sich ein neues Mitglied bei Ihnen anmelden.

sollten sie das sogar, zumindest die Passwörter.

Aus diesem Grund wurden die Skripte so parametrisiert, dass Sie sie sofort benutzen können, sobald Sie die MySQL-Daten, die zu Ihrer Umgebung passen eingegeben haben. Diese Anpassung müssen Sie nur in der Datei userdb.php3 vornehmen – alle anderen Skripte müssen nicht angepasst werden.

Den anzupassenden Bereich finden Sie ganz oben in userdb.php: siehe dazu Lis-

ting 1 im Kasten auf der übernächsten Seite.

Für `$SDB_NAME` tragen Sie den Namen Ihrer Datenbank ein. Als `$SDB_SERVER` tragen Sie den Namen Ihres MySQL-Servers ein. Das ist meist `localhost`, allerdings kann es sein, dass Sie hier einen echten Rechnernamen benötigen. Welcher das ist, erfahren Sie bei Ihrem Hosting-Provider.

`$SDB_USERNAME` und `$SDB_PASSWORD` sind der User-Name und das Passwort eines MySQL-Accounts mit Schreibrechten in dieser Datenbank. Diese Rechte werden im Skript benötigt, schließlich sollen neue User in der Tabelle eingetragen werden. Die Anwesenheit dieser Account-Informationen ist der hauptsächliche Grund dafür, dass Sie die Datei userdb.php in einem „sicheren“ Bereich auf Ihrem Web-Server ablegen, sodass nicht jedermann auf das Passwort und den User-Namen zugreifen kann.

## ■ Tabelle anlegen

Zur Aufnahme der Daten müssen Sie als erstes eine Tabelle anlegen. Dazu verwenden Sie, wie bereits erwähnt, am besten phpMyAdmin. Mit Hilfe dieses Programms richten Sie eine neue Tabelle mit dem Namen „user“ und fünf Feldern ein:

- id
- name
- password
- lastlogin
- level

`id` ist dabei vom Typ `int` – das sollte auch für eine große Menge von Mitgliedern ausreichen. `id` markieren Sie außerdem als *Not null* und *auto\_increment*. Dadurch stellen Sie sicher, dass diese `id`-Werte innerhalb der Tabelle immer eindeutig sind. Soweit möglich, wird später intern nicht mit den User-Namen, sondern immer nur mit dem `id`-Wert operiert. Aus diesem Grund weisen Sie MySQL außerdem an, das ID-Feld als primären Index zu behandeln.

`name` und `password` legen Sie einfach als `VARCHAR`s mit einer Länge von 16 Zeichen an: Dadurch können Ihre Anwender nur User-Namen verwenden, die maximal eine solche Länge ha-

ben. Wenn Sie es für sinnvoll halten auch längere Mitgliedernamen zuzulassen, müssen Sie diese Länge entsprechend anpassen. Nachdem auch nach dem User-Namen oft gesucht werden wird, etwa jedesmal, wenn sich ein Anwender anmelden möchte, sollten Sie auch für dieses Feld einen Index anlegen.

`lastlogin` ist einfach ein Feld vom Typ `DATE`. Darin wird immer das Datum des letzten Log-Ins eines Users abgelegt. Wenn Sie nicht nur das Datum, sondern auch die Uhrzeit erfassen möchten, können Sie statt eines `DATE`-Typs auch `DATETIME` oder einen der anderen Datum/Uhrzeit-Typen verwenden.

`level` enthält den User-Level des Anwenders. Nachdem gerade mal drei User-Level vorgesehen sind, reicht als Datentyp `tinyint` völlig aus.

Speichern Sie die Veränderung in Ihrer Datenbank. Nun wird es Zeit sich dem PHP-Programm zuzuwenden.

## ■ Userdb.php3 - die Basis-Funktionalität

In Userdb.php3 programmieren Sie im Wesentlichen eine Klasse sowie eine



**DAS PASSWORT WIRD** dem neuen Mitglied per E-Mail zugestellt.

handvoll hilfreicher Symbole. Nachdem die Symbole den geringsten Teil der Arbeit ausmachen, zunächst ein paar Erläuterungen dazu. In PHP können Sie die Laufzeitumgebung um eigene Symbole erweitern. Das funktioniert so, dass Sie einem String einen Wert zuordnen. Nun können Sie den String so verwenden, als wäre er ein Teil des Sprachumfanges.

Um ein eigenes Symbol zu definieren, verwenden Sie den `define()`-Befehl. Um zum Beispiel dem String „PI“ den Wert „4“ zuzuordnen, würden Sie folgenden Befehl verwenden:

```
define("PI", 4);
```

Danach können Sie PI einfach verwenden. Wenn Sie also zum Beispiel `print "PI"` verwenden, erhalten Sie als Resultat die Ausgabe des Wertes 4. `defines()` in PHP

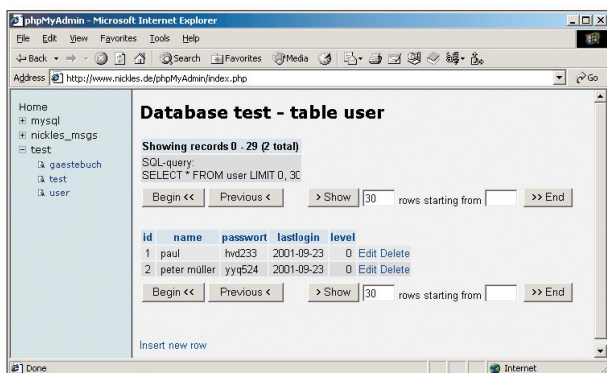


**WAR DIE ANMELDUNG** erfolgreich, erscheint eine entsprechende Meldung.

funktionieren in etwa so, wie `#define` in C/C++. Die Konstanten, die Sie für die User-Verwaltung definieren, dienen der späteren Identifizierung des User-Levels:

```
define( "USERLEVEL_GAST", 0);
define( "USERLEVEL_POWERUSER",
  1);
define( "USERLEVEL_ADMIN", 2);
```

Dadurch können Sie im Programm das Symbol `USERLEVEL_GAST` verwenden.



**NATÜRLICH KÖNNEN SIE IHRE MITGLIEDERVERWALTUNG** auch mit phpMyAdmin bearbeiten. Das ist auf Dauer allerdings etwas unpraktisch. Dazu nehmen Sie besser die Mitglieder-Verwaltung aus dem Beispiel-Programm.

den, wenn Sie einem User diesen Level zuweisen möchten. Das ist deutlich praktischer, als den numerischen Wert „0“ zu benutzen – der Text `USERLEVEL_GAST` ist sicherlich besser zu merken als die Bedeutung des Wertes „0“.

## ■ Die UserAdmin-Klasse

Nun beginnt die Implementierung einer Klasse. Dieser geben Sie den Namen `UserAdmin`. Diese Klasse übernimmt alle Aufgaben, die mit der Verwaltung der User zu tun haben. Sie können später eine Instanz von `UserAdmin` erzeugen, um irgendwelche Arbeiten mit den Anwenderdaten durchzuführen. `UserAdmin` erhält dazu die folgenden Funktionen: siehe dazu Listing 2 im Kasten auf der folgenden Seite.

Wenn Sie später weitere Funktionen für die Mitgliederverwaltung benötigen, ist die `UserAdmin`-Klasse ein guter Platz, um diese unterzubringen. Längerfristig wäre auch die Implementierung einer `User`-Klasse sinnvoll. Eine solche Klasse könnte alle Eigenschaften und Methoden, die auf einen User anwendbar sind, kapseln. Im Beispiel wurde das aus Gründen der Übersichtlichkeit nicht getan.

Praktisch alle Methoden des `UserAdmin` haben einen ähnlichen Aufbau. Zunächst wird eine Verbindung zum MySQL-Server aufgebaut. Dann wird ein SQL-Query String zusammengesetzt und die SQL-Abfrage durchgeführt. Schließlich wird das Ergebnis der Abfrage vom MySQL-Server abgeholt und bei Bedarf entsprechend weiterverarbeitet.

Hier beschreiben wir eine vollständige Funktion. (Siehe Listing 3 im Kasten auf der folgenden Seite.) Die übrigen

Funktionen des `UserAdmin` werden nur noch mit ihren speziellen Aufgaben und Unterschieden erläutert, ohne dass auf die immer wiederkehrenden Arbeitsschritte eingegangen wird.

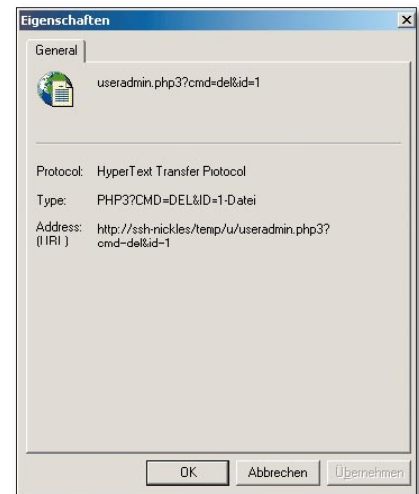
Die Funktion `IsDefined()` überprüft, ob ein gegebener User-Name (`$name`) bereits verwendet wird. Das brauchen Sie zum Beispiel, um sicherzustellen, dass keine doppelten User-

Namen auftreten.

Zunächst finden Sie in dieser Funktion einige „global“ Statements. Diese teilen PGP mit, dass die Funktion „globale“ Variable verwendet. Bei diesen Variablen handelt es sich um die zu Beginn des Skriptes von Ihnen anzupassenden Zugriffsdaten auf Ihren MySQL Server.

Nun wird mit `mysql_pconnect()` eine Verbindung zum angegebenen SQL-Server aufgebaut. Dabei muss nicht nur der Name des Servers, sondern auch ein gültiger Satz an

User-Name und Passwort verwendet werden. `mysql_pconnect()` unterscheidet sich von `mysql_connect()` dadurch, dass diese Funktion versucht, eine bereits bestehende Verbindung mit dem SQL-Server wiederzuverwenden. Das

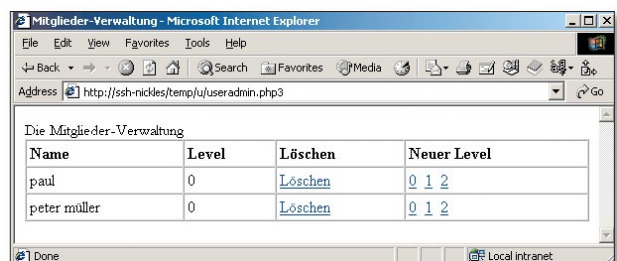


**DIE KOMMANDOS** für die Mitglieder-Verwaltung kodieren Sie direkt in die URLs in der Tabelle.

geht unter Umständen nicht nur schneller, sondern spart auch Ressourcen.

Im zweiten Schritt stellen Sie den SQL-Query String zusammen. Auch wenn die Bezeichnung Query-String darauf hinweist, diese Query-Strings sind keine Strings, die immer nur Abfragen enthalten. Statt dessen können Sie mit einem solchen String auch Daten aus der Datenbank entfernen, neue Tabellen anlegen und eine Vielzahl an weiteren Befehlen an Ihren MySQL-Server senden.

In diesem Fall handelt es sich beim Query-String um eine reine Abfrage. Erfragt wird, die Anzahl der IDs aller Mitglieder mit dem Namen, der in der Variable `$name` steht.



**MIT DER MITGLIEDERVERWALTUNG** löschen Sie Mitglieder oder verändern deren User-Level.

Diese Anfrage senden Sie mit `mysql_db_query()` an Ihr Datenbanksystem. Dabei geben Sie neben dem Query-



String auch den Namen der zu befragenden Datenbank und die Connector-ID zum Datenbankserver an.

Im nächsten Schritt erfragen Sie die Antwort auf Ihre Anfrage mit `mysql_fetch_object()`. Diese Funktion liefert Ihnen ein PHP-Objekt, das für jedes erfragte Feld eine Member-Variable hat. In diesem Fall gibt es genau eine, und die hat den Namen „count“, denn das war der Name, unter dem Sie das Ergebnis erfragt haben („as count“).

Nachdem es keine doppelten User-Namen geben kann, steht in dieser Variablen entweder eine 1 oder eine 0. Im Falle einer „1“ gibt es den erfragten Namen bereits und die Funktion liefert `true`. Im anderen Fall ist der Name noch

nicht definiert und die Funktion liefert `false`.

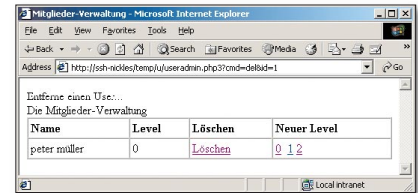
Wie bereits erwähnt funktionieren die übrigen Funktionen des UserAdmins ähnlich. Daher folgt nun der Schnelldurchlauf zur Funktionserklärung bei dem nur auf diese wesentlichen Merkmale der Funktion eingegangen wird.

- **AddUser()** - Mit dieser Funktion fügen Sie ein neues Mitglied zu Ihrer Datenbank hinzu. AddUser() erhält zwei Parameter: Den Usernamen und das zugehörige Passwort.

- **Sqs** = „insert into user ( name, password, lastlogin, level) values ('\$user', '\$password', NOW(), USERLEVEL\_GAST)“;

Bei SQL fügen Sie einen Datensatz mit

dem `insert into`-Befehl hinzu. Dieser Befehl wird gefolgt von der Tabelle, in der die neuen Daten eingetragen werden sollen, in diesem Fall also `user`. Danach ge-



**DIE EINZELNEN FUNKTIONEN** der User-Verwaltung geben kurze Meldungen oben auf der Seite aus.

ben Sie in runde Klammern gefasst an, welche der Tabellenfelder mit Daten belegt werden sollen.

Beim Hinzufügen eines neuen Mitglieds sind das alle Felder bis auf das `id`-Feld, das von MySQL automatisch hochgezählt und gesetzt wird, weil Sie dies bei der Definition der Tabelle so vorgesehen haben.

Nun folgen die Werte nach dem Kennwort `values`, ebenfalls in Klammern eingefasst. Dabei ist es wichtig, dass Sie Text-Variable mit Hochhaken umfassen, sonst kommt SQL durcheinander, wenn die Variablen-Inhalte auch Leerzeichen enthalten. Die Werte müssen in der gleichen Reihenfolge aufgelistet werden, in der Sie zuvor die Feldnamen angegeben haben. Auf diese Weise laden die Daten auch in den gewünschten Feldern.

Als Werte verwenden Sie die Parameter `$user` und `$password`, die der Funktion übergeben wurden. Als Datum für das letzte Log-in können Sie die MySQL-Funktion `NOW()` verwenden.

Wenn MySQL den Query-String auswertet, wird es statt des Funktionsnamens den Rückgabewert dieser SQL-Funktion verwenden, und der ist in diesem Fall das aktuelle Datum.

Schließlich geben Sie noch den User-Level an. Dieser ist `SERLEVEL_GAST`. Hier verwenden Sie zum ersten Mal das weiter oben in der Datei definierte eigene Symbol: Alle Mitglieder sind also von Haus aus zunächst nur mit einem „Gast“-Account ausgestattet. Soll sich diese Status ändern, müssen Sie das mit einer anderen Funktion nachträglich tun.

## CreatePassword()

An dieser Stelle stellt sich die Frage, woher das Passwort zum User-Account stammt. Hier gibt es eine ganze Reihe

### LISTING 1

```
// ***** BEGINN DER ANPASSUNG *****
$DB_NAME      = "";
$DB_USERNAME  = "";
$DB_PASSWORD  = "";
$DB_SERVER    = "";
// ***** ENDE DER ANPASSUNG
```

### LISTING 2

```
class UserAdmin
{
    function IsDefined( $name);
    function AddUser( $user, $password);
    function CreatePassword();
    function GetUsers();
    function DeleteUser( $idUser);
    function SetUserLevel( $idUser, $idLevel);
    function SetLastLogin( $idUser);
    function CheckUser( $name, $password);
    function GetUserData( $idUser, &$lastlogin, &$level);
}
```

### LISTING 3

```
function IsDefined( $name)
{
    global $DB_SERVER;
    global $DB_USERNAME;
    global $DB_PASSWORD;
    global $DB_NAME;

    $c = mysql_pconnect( $DB_SERVER, $DB_USERNAME, $DB_PASSWORD);
    $qs = "select count(id) as count from user where name='$name'";
    $r = mysql_db_query( $DB_NAME, $qs, $c);
    $o = mysql_fetch_object( $r);
    if( $o->count == 1) return true;
    return false;
}
```

### LISTING 4

```
function CreatePassword()
{
    srand((double)microtime()*1000000);
    $str = sprintf("%c%c%c%c%c",
        (rand()%26 + 97),
        (rand()%26 + 97),
        (rand()%26 + 97),
        (rand()%10 + 48),
        (rand()%10 + 48),
        (rand()%10 + 48) );
    return $str;
}
```

von Verfahrensweisen. Zum Beispiel könnte man, dem Mitglied die Möglichkeit geben, sein Passwort selbst festzulegen.

Viel weniger Aufwand ist es, wenn Sie das Passwort selbst erzeugen und genau das tut *CreatePassword()*. Die Funktion erzeugt ein Passwort, das sich aus drei Buchstaben und drei Nummern zusammensetzt.

Dazu wird sechsmal die Funktion *rand()* aufgerufen und die Ergebnisse dieser Aufrufe werden in einem String zusammengesetzt. Siehe dazu Listing 4 im Kasten auf der vorherigen Seite.

Damit *rand()* funktioniert, müssen Sie den Zufallszahlengenerator zuvor mit einem Aufruf von *srand()* initialisieren. Der Generator erhält damit den „Seed“, der sicherstellt, dass mehrere Aufrufe von *rand()* tatsächlich unterschiedliche Resultate liefern.

Nachdem der String für das Passwort so zusammengesetzt wurde, liefert die Funktion ihn zurück.

## ■ *GetUsers()* – alle Userdaten ermitteln

Die Funktion *GetUsers()* liefert ein Array aus PHP-Objekten zurück. Jedes Objekt entspricht dabei einem Mitglied und enthält dessen Namen, *userlevel* und *id*. Das geht folgendermaßen:

Zunächst brauchen Sie den passenden Query-String und senden die Anfrage an den Server. Das Zusammensetzen solcher Strings sind Sie mittlerweile gewohnt:

```
$qs = "select
  ↳ id,name,level from
  ↳ user order by id";

$r = mysql_db_query(
  ↳ $DB_NAME, $qs, $c);
```

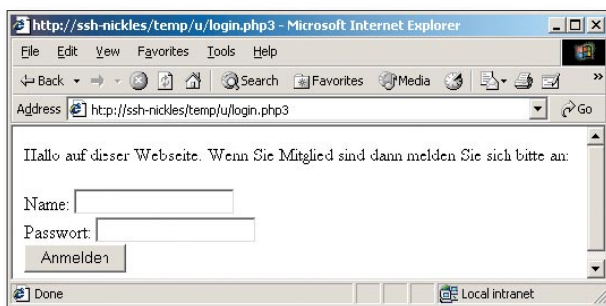
Nun programmieren Sie eine Schleife, die so lange läuft, wie Sie Objekte mit *mysql\_fetch\_object()* zurückerhalten. Jeder Aufruf von *mysql\_fetch\_object()* liefert Ihnen dabei den jeweils nächsten Teil der Ergebnisse zurück. Sie erhalten im

Wesentlichen pro Aufruf von *mysql\_fetch\_object()* einen Satz Mitgliedsdaten.

```
(while( $o = mysql_fetch_object(
  ↳ $r))

  $arrUsers[] = $o;
```

Die so gelieferten Objekte speichern Sie in einem PHP-Array. Diese Arrays zeichnen sich dadurch aus, dass es besonders einfach ist, hinten Elemente anzuhängen: Alles was Sie tun müssen, ist bei der Zuweisung ans Array, keinen Index anzugeben. Das bedeutet, wenn Sie



**MIT DEM ANMELDEFORMULAR** kann sich Ihr Mitglied bei der Webseite anmelden.

keine Zahl in die eckigen Klammern eintragen, lädt das „neue“ Element automatisch hinten am Array und das Array wird automatisch vergrößert.

Dieses Array liefern Sie mit *return \$arrUsers* zurück an die aufrufende Funktion.

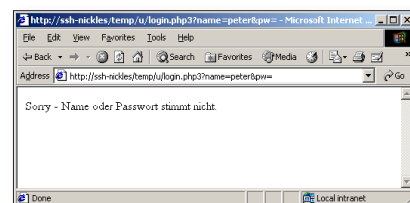
## ■ Mitglieder löschen

Mit *DeleteUser(\$idUser)* löschen Sie ein Mitglied aus der Tabelle der Mitglieder:

```
$qs = "delete from user where
  ↳ id=$idUser";
```

Das Löschen von Datensätzen erfolgt mit dem *delete*-Befehl. Dabei müssen Sie noch die Tabelle angeben, aus der

gelöscht werden soll, also in diesem Fall *from user*. Schließlich müssen Sie noch ein Kriterium für den Löschvorgang angeben. In diesem Fall sollen alle Einträge gelöscht werden deren „Id“ den Wert aus *SidUser* haben. Nachdem die User-Id ein eindeutiger Wert ist, handelt es sich dabei immer um maximal ein Mitglied.



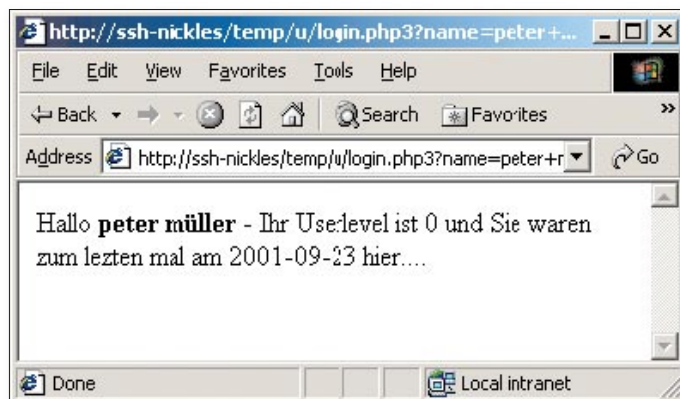
**MIT FALSCHEN USER-NAMEN** oder Passwörtern kann man sich natürlich nicht anmelden.

An dieser Stelle ein Wort zum Thema Vorsicht. MySQL ist nicht besonders rücksichtsvoll, was Ihre Query-Strings angeht. Wenn Sie MySQL ein gültiges Query senden, wird das auch ausgeführt. Es gibt danach kein zurück mehr. Das ist beim *delete*-Befehl naturgegeben besonders gefährlich. So wäre zum Beispiel der Query-String

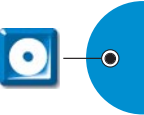
```
$qs = „delete from user“
```

sehr wohl ein gültiges Query. Nachdem keine Bedingung für das Löschen angegeben wurde, tut MySQL das Naheliegende: Es löscht einfach alle Einträge aus der angegebenen Tabelle. Das ist schlimm, besonders wenn Sie nicht mehr beim Testen sind und schon Hunderte oder Tausende von Mitgliedern in Ihrer Tabelle aufgenommen hatten. Nachdem solch ein Fehler beim Programmieren im Eifer des Gefechts leicht passieren kann, hier nochmals der Hinweis: Wenn Sie einen „gefährlichen“ SQL-Befehl wie *delete* verwenden, schauen Sie vor dem Testen Ihres Skriptes besser dreimal nach, ob Sie auch wirklich das löschen, was Sie auch wirklich löschen wollen.

Das können Sie zum Beispiel dadurch testen, indem Sie vor dem ersten Durchlauf Ihres Programms mit *delete*, diesen Befehl durch *select \** ersetzen. Dann löscht SQL nichts, sondern liefert statt dessen alles, was es löschen würde als Ergebnis zurück. Als Safety-Check ist dieses Vorgehen recht sinnvoll.



**DIE BEGRÜSSUNGSMELDUNG** für einen angemeldeten User könnte man sicherlich noch verschönern. Funktional ist aber alles Benötigte vorhanden.



## Mitglieder-Daten verändern

Mit den Funktionen `SetUserLevel()` und `SetLastLogin()` verändern Sie das User-Level und setzen den Zeitpunkt des letzten Log-ins. Beide Funktionen verwenden den gleichen Mechanismus, daher finden Sie hier nur eine exemplarische Funktion im Auszug:

```
$qs = "update user set
level=$idLevel where id=
$idUser";
```

Beim Update-Befehl von MySQL geben Sie, wie bereits von anderen Befehlen gewohnt, die zu verändernde Tabelle an. Dann legen Sie mit `set` fest, welches Feld oder welche Felder Sie verändern wollen und auf welchen Wert das oder die Felder gesetzt werden sollen. Schließlich bekommt der Befehl noch eine Bedingung mitgeliefert, die festlegt welche Daten verändert werden sollen. Hier gelten die gleichen Vorsichtsmaßnahmen wie zuvor beim `delete`-Befehl!

## User checken – passt das Passwort?

Die Funktion `CheckUser()` erfüllt einen der elementaren Zwecke der Mitglieder-Datenbank: Sie überprüft ob ein gegebenes Passwort zu einem Anwendernamen passt:

```
$qs = "select id from user where
name='$name' and password=
'$password'";
```

Das geht ähnlich wie bei der Funktion `IsDefined`, allerdings wurde hier aus Demonstrationsgründen nicht der `count()`-Befehl verwendet, sondern eine Alternative implementiert.

```
$r = mysql_db_query( $DB_NAME,
$qs, $c);
```

```
$scr = mysql_num_rows( $r);
```

Nach dem Absenden der Anfrage können Sie die Anzahl der gelieferten Zeilen aus der Tabelle erfragen. Wenn Sie wissen, dass es sich beim Resultat immer nur um eine oder keine Spalte handeln kann, hat das keine weiteren dramatischen Implikationen.

Ansonsten ist es wenig sinnvoll einen großen Satz an Ergebnissen zu erfragen, nur um dann dessen Größe zu berechnen, wenn Sie mit dem `count()`-Befehl diesen Wert auch direkt erfragen können:

```
if( ! $scr) return -1;
$o = mysql_fetch_object( $r);
return $o->id;
```

In diesem Fall benötigen Sie sogar ein Element des Resultats. `CheckUser()` liefert die ID des Users zurück, wenn das Passwort zum Namen passte. Mit dieser ID können dann andere Funktionen

des letzten Anmeldens und der Userlevel: siehe dazu Listing 5 im Kasten unten.

An dieser Stelle sehen Sie zum ersten Mal eine PHP-Funktion bei der *Call by*

```
ssh-nickles - PuTTY
F10 key ==> File Edit Search Buffers Windows System Help
// userdaten erfragen
function GetUserData( $idUser, &$lastlogin, &$level)
{
    global $DB_SERVER;
    global $DB_USERNAME;
    global $DB_PASSWORD;
    global $DB_NAME;

    $c = mysql_pconnect( $DB_SERVER, $DB_USERNAME, $DB_PASSWORD);
    $qs = "select lastlogin,level from user where id=$idUser";
    $r = mysql_db_query( $DB_NAME, $qs, $c);
    $o = mysql_fetch_object( $r);
    $lastlogin = $o->lastlogin;
    $level = $o->level;
}
?>
```

----- C:\Jed B0.99.9) Emacs: userdb.php (Text) 156/173 3:57pm -----

**AUCH PHP KENNT *call by Reference*.** Soll ein Parameter *by Reference* übergeben werden statt *by value*, müssen Sie nur einen Ampersand davor schreiben.

weiterarbeiten, wie Sie im Verlaufe des Beitrages noch sehen werden.

Stimmte das Passwort nicht (oder ist der User-Name unbekannt), liefert `CheckUser` „-1“ zurück.

## User-Daten erfragen

Schließlich bietet der UserAdmin noch eine Klasse zum Erfragen von Mitgliederdaten. Im Gegensatz zu `GetUsers()`, wo Daten über alle Mitglieder ermittelt werden, liefert `GetUserData()` nur die Daten zu einem bestimmten Mitglied. Geliefert werden dabei der Zeitpunkt

*Reference* statt *Call by Value* verwendet wird. Das passiert für die Variable `$lastlogin` und `$level`. *Call By Reference* benötigen Sie, wenn Sie einen Rückgabewert in einen Parameter liefern möchten.

Dabei wird der Wert der Variablen nicht nur lokal, sondern auch für die aufrufende Funktion verändert. In C/C++ entspräche das einer Funktion, der ein Parameter *by Reference* oder per Zeiger übergeben wird. Um anzuzeigen, dass Sie *Call By Reference* verwenden wollen, müssen Sie in der Ar-

### LISTING 5

```
function GetUserData( $idUser, &$lastlogin, &$level)
{
    $c = mysql_pconnect( $DB_SERVER, $DB_USERNAME, $DB_PASSWORD);
    $qs = "select lastlogin,level from user where id=$idUser";
    $r = mysql_db_query( $DB_NAME, $qs, $c);
    $o = mysql_fetch_object( $r);
    $lastlogin = $o->lastlogin;
    $level = $o->level;
}
```

### LISTING 6

```
if( IsSet( $name))
{
    require("userdb.php");
    $ua = new UserAdmin;
    $id = $ua->CheckUser( $name, $pw);
    if( $id == -1)
    {
        print 'Sorry - Name oder Passwort stimmt nicht.';
    }
    else
    {
        $ua->GetUserData( $id, $lastlogin, $level);
        print "Hallo <b>$name</b> - Ihr Userlevel ist $level und Sie waren
        zum letzten mal am $lastlogin hier....<br>";
    }
}
```

gumentenliste nur einen Ampersand vor die Variable schreiben. Mehr ist nicht notwendig.

Soviel zum UserAdmin – Sie haben nun alle Funktionen für eine Mitglieder-verwaltung zur Verfügung. Einige Beispiele für die Anwendung in der Klasse folgen nun im Rest dieses Beitrages.

## Mitgliederanmeldung

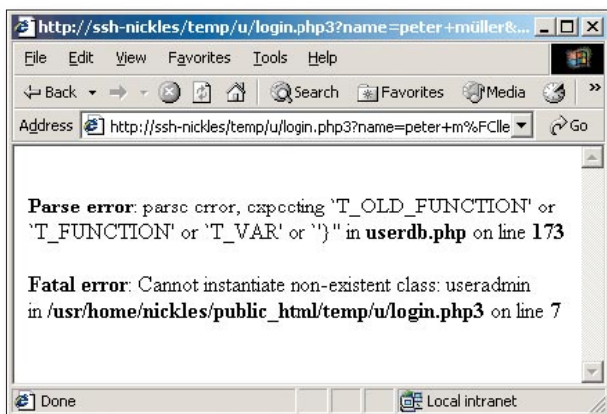
Das einfachste Beispiel finden Sie in der Datei „login.php3“. Hier wird zunächst ein Formular zum Anmelden angezeigt. Füllt ein Surfer das Formular aus und schickt es ab, werden die Formular-Angaben ausgewertet: siehe dazu Listing 6 im Kasten auf der vorherigen Seite.

Zunächst inkludieren Sie die Datei „userdb.php“. Dadurch steht Ihnen der UserAdmin zur Verfügung. Um diesen benutzen zu können, benötigen Sie zunächst eine Instanz dieser Klasse. Diese erzeugen Sie mit `new UserAdmin` und speichern sie in der Variablen `$ua`.

Dann können Sie mit `$ua->CheckUser( $name, $password)` überprüfen, ob der User-Name existiert und das Passwort zum Namen passt. Die `CheckUser()`-Funktion wurde bereits weiter oben erläutert.

Passen die Angaben nicht, liefert `CheckUser()` -1, und das Programm gibt eine entsprechende Meldung von

sich. Sind die Angaben aber in Ordnung, erfragen Sie mit `$ua->GetUserData()` unter Verwendung der zuvor ermittelten User-ID die Daten des Mitglieds und können es dann mit seinem User-Level



**FEHLER IM QUELL-CODE** mag PHP nicht besonders gern. Allerdings stimmen die Fehlermeldungen auch nicht immer: Hier fehlte eine schließende Klammer.

und dem Zeitpunkt des letzten Log-ins begrüßen.

## Neue Mitglieder aufnehmen

In der Datei „user.php“ finden Sie ein Beispiel für das Formular zum Aufnehmen eines neuen Mitglieds. Das neue Mitglied gibt dazu den gewünschten User-Namen und seine E-Mail-Adresse an. Damit beginnen Sie:

```
function InsertUser( $name,
    $email)
{
    $ua = new UserAdmin;
    if( $ua->IsDefined( $name))
        print 'Sorry, dieser Name
        ist bereits vergeben<br>';
```

Wie bereits gewohnt benötigen Sie zunächst eine Instanz des UserAdmins. Die verwenden Sie zunächst, um herauszufinden, ob der gewünschte User-Name bereits vergeben wurde oder nicht.

Das tun Sie mit der `IsDefined()`-Methode. Existiert der Name bereits, geben Sie eine entsprechende Meldung aus. Klüger wäre es hier unter Umständen, wenn Sie statt dessen einen Namen synthetisieren und diesen dann verwenden. So bekommt das Mitglied auf jeden Fall einen Benutzernamen und kann weiterarbeiten. Allerdings wäre es in diesem Fall notwendig, den Mitgliedern eine Möglichkeit zu geben, ihren User-Namen später zu verändern.

War der Name hingegen noch nicht vergeben, erzeugen Sie mit `$ua->CreatePassword()` ein neues Passwort:

```
$pw = $ua->CreatePassword();
Mit User-Name und Passwort können Sie nun das neue Mitglied in der Datenbank eintragen, und den Surfer über diesen Umstand informieren:
$ua->AddUser( $name, $pw);
print 'Hurra! - Ein neues
Mitglied.<br>';
```

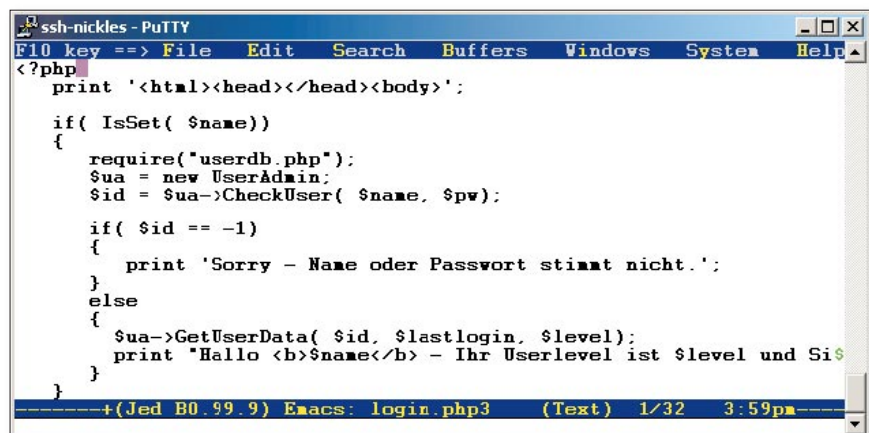
Schließlich müssen Sie dem neuen Mitglied noch sein Passwort mitteilen. Das würde auch direkt über die Webseite funktionieren. So etwas hat aber diverse Implikationen, nicht zuletzt sicherheitsrelevante. Daher ist es besser das Passwort per E-Mail zuzustellen, und das geht in PHP mit der `mail()`-Funktion.

```
$body = "Hallo! - Ihr Passwort
lautet; $pw";
mail( $email, "Ihr Mitglieds-
Passwort", $body, "From:
useradmin@this_site.com");
```

Ein letztes, etwas umfangreicheres Beispiel finden Sie im Programm „useradmin.php“. Dort befindet sich eine einfache Benutzerverwaltung. Diese stellt Ihnen eine Liste aller Mitglieder auf einer Webseite zur Verfügung. Die Mitglieder können dabei gelöscht werden und auch ein Wechsel der User-Level ist möglich.

In diesem Beitrag haben Sie erfahren, wie Sie eine Mitgliederverwaltung mit PHP und MySQL implementieren können. Die vorliegenden Beispiel-Quel-Codes sollten für die erste eigene Verwaltung mehr als ausreichend sein. Trotzdem haben Sie natürlich jede Menge Spielraum für Erweiterungen.

UR



**WENN SIE EINEN INTERAKTIVEN ACCOUNT** besitzen, ist das Editieren von Dateien auf dem Server mit JED sehr angenehm. Probieren Sie diesen Editor mal aus!