Datenbankanbindung mit mySQL und PHP

Sabine Grötz

1 Datenbankanbindung

1.1 Dynamische Inhalte im Web

In der Anfangszeit des Web wurden nur statische Inhalte zur Verfügung gestellt. Sollte der Inhalt geändert werden, musste man manuell die Änderungen vornehmen, was vor allem mühsam und unübersichtlich ist. Seit es Werkzeuge wie Perl oder PHP gibt, können Inhalte auf Web-Seiten auch dynamisch generiert werden. Sie bieten die Möglichkeit, mehr oder weniger komplexe Programmstrukturen zu implementieren. Durch entsprechende Anweisungen ist auch die Kopplung an eine Datenbank möglich. Somit lassen sich völlig neue Anwendungen verwirklichen, wie z.B. die dynamische Bereitstellung von gespeicherten Informationen und deren interaktive Änderung über das Web.

Eine einfache Datenbank auf dem Server zu haben reicht aber nicht. Um den Zugriff von der HTML-Seite her zu ermöglichen, bedarf es mindestens 2 Applikationen: der Web-Server, welcher die HTML-Seiten bereitstellt, und der Datenbank-Server, welcher den Zugriff auf die Daten ermöglicht. Zusätzlich wird dabei oft eine dazwischenliegende Applikation verwendet, das Web/Datenbank-Gateway, welches die Anfrage der HTML-Seite entgegennimmt und sie so aufbereitet, dass sie vom Datenbank-Server verstanden wird.

1.2 Zusammenspiel von Browser, Webserver und Datenbankserver

Zwischen einer statischen und einer dynamisch generierten Website merkt der Anwender normalerweise keinen Unterschied, außer vielleicht an der angezeigten URL. Klickt er auf einen Link, so fordert der Browser die mit dem Link spezifizierte URL vom Webserver an. Der Webserver erkennt z.B. an der Dateiendung, dass es sich hierbei um eine noch zu generierende Seite handelt. Die zu generierende Seite wird, zusammen mit verfügbaren Informationen, an den Interpreter übergeben.

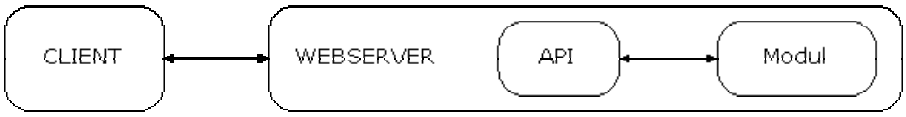
Der Interpreter ist entweder als externes Programm implementiert, oder in den eigentlichen Server fest integriert. Er filtert dann die für ihn bestimmten Anweisungs-Blöcke heraus und führt sie aus. Beim Ausführen können wiederum andere Programme aufgerufen werden, unter anderem ist dies auch die Stelle für etwaige Datenbankabfragen. Zugriffe auf eine Datenbank können mit entsprechenden Modulen oder über eine ODBC-Schnittstelle erfolgen. Hat der Interpreter die endgültige, in HTML vorliegende Fassung erzeugt, gibt er diese an den Webserver zurück, der sie an den Browser schickt.

Zwischen Datenbank- und Web-Server vermittelt ein so genanntes Web/Datenbank-Gateway. Diese Aufgabenteilung ergibt sich aus den unterschiedlichen Fähigkeiten der beiden Servertypen: Datenbankmanagementsysteme sind darauf spezialisiert, strukturierte Information zu verwalten und auf beliebige Anfragen dynamisch Treffermengen zu generieren. Web-Server hingegen verteilen in erster Linie statische HTML-Dokumente.

1.3 CGI und API

Die einzige Möglichkeit, von einem Web-Server aus einen anderen Dienst anzubieten, besteht darin, eine externe Anwendung damit zu beauftragen. Als Schnittstelle nach außen hat der Web-Server 2 Möglichkeiten: CGI und API.

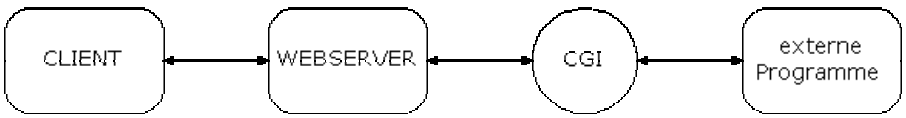
1.3.1 CGI (*Common Gateway Interface*)
CGI ist eine normierte Schnittstelle zwischen Webserver und einem Softwaremodul (externes Programm), d.h. es beschreibt, wie der Webserver ein Programm bzw. einen Interpreter aufrufen soll und wie die Daten wieder zurückgeschickt werden sollen.

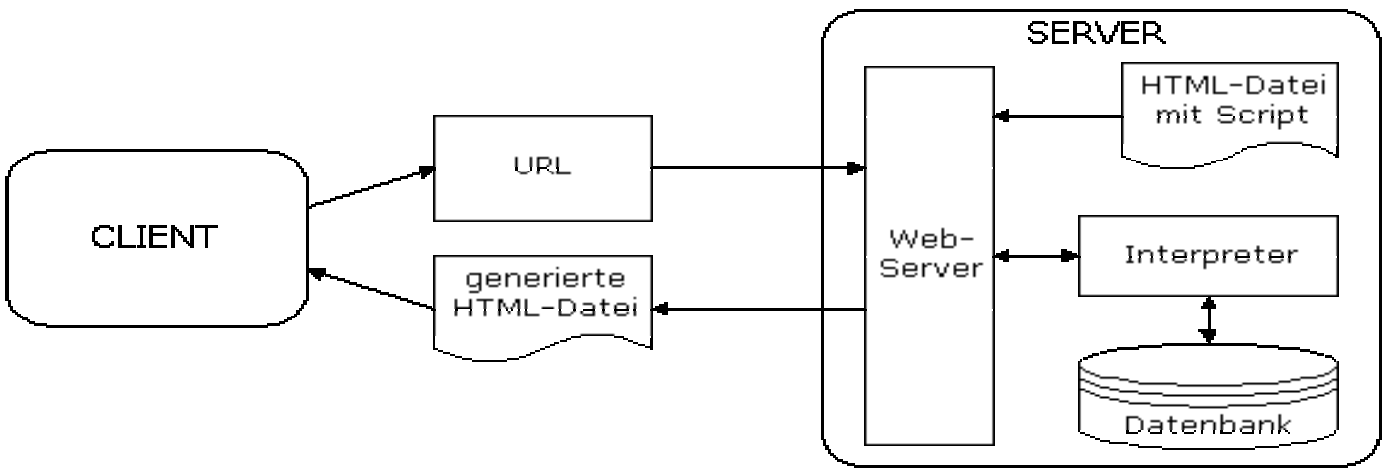


1.3.2 API (*application programming interface*)
API sind Schnittstellen mit denen Webserver erweitert werden können. Die Module (z.B. Interpreter) werden Bestandteil des Servers selbst. Dadurch kann auch eine Geschwindigkeitssteigerung erreicht werden.

1.4 *Datenbank-Management-System* DBMS

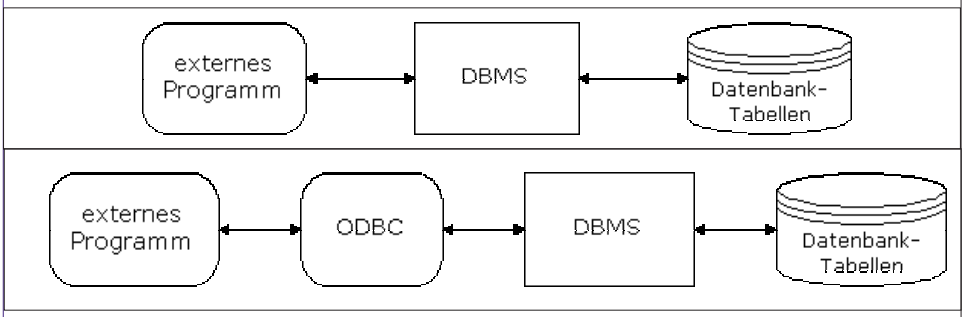
Eine Datenbank ist eine systematische Sammlung von Daten. Sinn und Zweck einer solchen Sammlung ist jedoch nicht die Anhäufung der Daten alleine, es geht vielmehr darum, den Zugriff auf verfügbare Daten zu erleichtern bzw. erst zu ermöglichen. Hierzu ist es notwendig, dass diese Daten organisiert und verwaltet werden. Zur Nutzung und Verwaltung der in der Datenbank gespeicherten Daten benötigt der Anwender ein Daten-





http://www.tierschutz.at/

bank-Management-System (DBMS, engl. *Data Base Management System*). Das DBMS muss Abfragen durchführen und Daten manipulieren können und für mehrere Benutzer gleichzeitig zur Verfügung stehen (***Bild oben***).



Ursprünglich waren alle DBMS mit einer nicht-standardisierten Programmierschnittstelle ausgestattet. Für den einheitlichen Datenbankzugriff hat Microsoft einen Standard etabliert: ODBC (*Open Database Connectivity*). Er ermöglicht über eine einheitliche Schnittstelle den Zugriff auf Datenbanken, die ihrerseits über einen ODBC-Treiber verfügen (dazu gehören alle wichtigen am Markt vertretenen Datenbanken z.B. dBase, Oracle, DB/2, SQL-Server, Access, etc.).(***Bild unten***)

Webserver und Datenbank-Server (DBMS und Datenbank-Tabellen) müssen nicht dieselben physikalischen Rechner sein. Der Datenbank-Server und der Webserver können auch über ein Netzwerk kommunizieren, d.h. auch übers Internet.

2 SQL

2.1 Was ist SQL?

SQL (*Structured Query Language*) ist eine standardisierte Definitions- und Abfragesprache für relationale Datenbanken zur Kommunikation mit einem Datenbank-Management-System und wurde wurde Ende der 70er Jahre von IBM entwickelt. Sie ermöglicht einerseits Datenabfragen und andererseits Datenmanipulation. Trotz Normung haben einzelne Hersteller ihre SQL-Dialekte individuell erweitert.

SQL kann nicht allein stehend, sondern nur in Verbindung mit einer anderen Programmiersprache oder einem speziellen Datenbankprogramm verwendet werden. Weiterhin ist SQL keine prozedurale Computersprache wie BASIC, C oder PASCAL. Mit einem SQL-Befehl wird nicht definiert wie der Computer die Aufgabe erfüllen soll sondern man definiert nur was man gerne tun möchte. Das DBMS nimmt den Befehl entgegen und bestimmt dann die Art und Weise, wie die Aufgabe möglichst effizient erfüllt werden soll. Mit SQL kann man sowohl ad hoc Datenbankanfragen durchführen als auch Programme für immer wiederkehrende Routinen schreiben. SQL-Befehle gliedern sich in zwei Bereiche der Datendefinitionssprache (DDL) und der Datenmanipulationssprache (DML), aber dazu später.

2.2 Exkurs: Aufbau einer relationalen Datenbank

Relationale DBMS speichern Daten in Tabellen (auch Relationen genannt). Eine Tabelle besteht aus Zeilen (=Datensätze) und Spalten (=Attribute). Dadurch sind relationale Datenbanken enorm flexibel, z.B. können Daten in verschiedenen Tabellen auf verschiedene Arten miteinander verknüpft werden, um Informationen zu erhalten. Um eine Beziehung zwischen zwei Tabellen herzustellen, werden die Tabellen über ein gemeinsames Datenfeld verbunden. Die übereinstimmenden Werte bilden den so genannten Primär- bzw. Fremd-Schlüssel. Betrachten wir diese Definitionen an einem Beispiel:

Tabelle Tierfamilie

ID	Tierfamilie
1	Säugetiere
2	Fische
3	Reptilien
4	Vögel

Tabelle Tierart

ID	Tierfamilie_ID	Tierart
11	1	Hunde
12	1	Katzen
13	1	Nagetiere
14	2	Goldfisch

In der Abbildung werden zwei Tabellen dargestellt: **Tierfamilie** und **Tierart**. Die Zeilen repräsentieren untereinander gleichartige Informationseinheiten. Diese Datensätze sind gegliedert in Felder (Attribute). Die Spalten der Tabellen enthalten gleichartige Felder der Datensätze. Sie sind mit den Namen der Felder überschrieben (**ID**, **Tierfamilie**, **Tierart**).

Jede der Tabellen hat verschiedene Datensätze, die durch einen eindeutigen Schlüssel (Feld **ID**) gekennzeichnet sind, welcher zur Identifikation des Datensatzes dient. In der Tabelle **Tierart** gibt es ein weiteres Feld mit dem Namen **Tierfamilie_ID**, welches die Tabelle **Tierart** mit der Tabelle **Tierfamilie** verknüpft. Das Feld **Tierfamilie_ID** ist ein Fremdschlüssel.

Jedes Feld besitzt einen Datentyp, der innerhalb einer Spalte gleich sein muss. Die wichtigsten Datentypen sind:

- Zahlen (z.B. ganzzahlig = `INTEGER`, Fließkomma = `FLOAT`)
- Text (fixe Länge = `CHAR(m)`, variable Länge = `VARCHAR(n)`)
- Datum und Uhrzeit (z.B. Datum = `DATE`, Uhrzeit = `TIME`, beides = `TIMESTAMP`)

In obigen Beispiel wären die Schlüsselfelder `INTEGER`-Felder und die Bezeichnungsfelder (**Tierfamilie** und **Tierart**) Textfelder (`CHAR` oder `VARCHAR`).

2.3 DDL: *Data Definition Language*

Anweisungen zur Datendefinition dienen zur Einrichtung, Änderung oder Löschung von Datenbankobjekten wie z. B. Tabellen und Indizes. Damit wird die gesamte Datenstruktur bzw. das physische Datenmodell definiert. Unter die DDL fallen folgende Statements:

- `CREATE` (Erzeugen neuer Objekte)
- `ALTER` (Verändern der Eigenschaften existierender Objekte)
- `DROP` (Löschen existierender Objekte)

2.3.1 `CREATE`-Statement

Mit dem `CREATE`-Statement kann man Datenbanken, Tabellen und Indizes anlegen.

Datenbank anlegen

```
CREATE DATABASE database_name
```

*Datenbank **tierschutz** anlegen*

```
CREATE DATABASE tierschutz;
```

Tabelle anlegen

```
CREATE TABLE table_name (create_definition,
[create_definition],…)
create definition:
column_name datatyp [NOT NULL] [UNIQUE]
[PRIMARY KEY]
```

Im `CREATE TABLE`-Statement dürfen mehrere `create_definition` durch Komma getrennt angegeben werden. Datentyp können z.B. `INTEGER`, `FLOAT`, `CHAR(m)`, `VARCHAR(n)` etc. sein. In eckiger Klammer stehen optionale Felder. Diese können angegeben werden oder auch nicht. "`NOT NULL`" bedeutet, dass der Feldinhalt nicht leer sein darf, "`UNIQUE`" bedeutet, dass der Feldinhalt eindeutig sein muss und "`PRIMARY KEY`" bedeutet, dass es sich beim angegeben Feld um einen Primärschlüssel handelt.

*Tabelle **tierart** anlegen mit den Feldern **id, tierfamilie_id** und **tierart***

```
CREATE TABLE tierart (id INTEGER PRIMARY KEY
UNIQUE NOT NULL, tierfamilie_id INTEGER NOT
NULL, tierart VARCHAR(255));
```

Index anlegen

```
CREATE INDEX index_name ON table_name
(column_name [, column_name]...)
```

Ein oder mehrere Felder können indiziert sein, d.h. die Datenbank besitzt für diese Felder einen Index, das ist ein sortiertes Verzeichnis der Datenwerte und der zu-

geordneten Datensätze (sowie ein Stichwortverzeichnis in einem Buch). Ein Index erleichtert der Datenbank das Finden von Datensätzen mit bestimmten Datenwerten. Für die Benutzer ist der Index unsichtbar.

Index tierart_index anlegen

```
CREATE INDEX tierart_index ON tierart (id,
tierfamilie_id);
```

2.3.2 ALTER-Statement

```
ALTER TABLE table_name {ADD| MODIFY|DROP}
(create_definition, [create_definition],…)
```

Mit dem `ALTER TABLE`-Statement kann die Datenstruktur einer existierenden Tabelle geändert werden. Es können neue Felder definiert werden, bestehende Felder können geändert oder gelöscht werden.

*Beispiel: neues Feld in Tabelle **tierfamilie** hinzufügen*

```
ALTER TABLE tierfamilie ADD latein_name
VARCHAR(255);
```

*Beispiel: bestehendes Feld **latein_name** in Tabelle **tierfamilie** ändern*

```
ALTER TABLE tierfamilie MODIFY latein_name
CHAR(50);
```

*Beispiel: Feld **latein_name** in Tabelle **tierfamilie** löschen*

```
ALTER TABLE tierfamilie DROP latein_name;
```

2.3.3 DROP-Statement

```
DROP DATABASE database_name
DROP TABLE table_name
DROP INDEX index_name ON table_name
```

Um erstellte Datenbanken, Tabellen oder Indizes zu löschen, verwendet man das `DROP`-Statement.

Löschen einer Datenbank

```
DROP DATABASE database_name;
```

Löschen einer Tabelle

```
DROP TABLE tierart;
```

Löschen eines Index

```
DROP INDEX tierart_index ON tierart;
```

2.4 DML: *Data Manipulation Language*

Mit der DML werden Informationen aus der Datenbank abgefragt und Zustände der Daten verändert. Zur Datenabfrage dient das `SELECT`-Statement. Zur Datenmanipulation dienen die `DELETE`-, `INSERT`- und `UPDATE`-Statements.

2.4.1 SELECT-Statement

Das `Select`-Statement dient zur Abfrage der Tabellen. Ein wichtiges Konzept von SQL ist es, dass Antworten auf Anfragen wieder Tabellen sind. Im Extremfall kann diese Tabelle auch nur eine Spalte und eine Zeile haben. Dieses Konzept erlaubt verschachtelte Anfragen, da ja das Ergebnis einer Anfrage wieder als Suchtabelle einer nachfolgenden Anfrage verwendet werden kann. Weiters können Aggregatsfunktionen (`Minimum`, `Maximum`, `Average`), mathematische und boolsche Funktionen (Addition, Subtraktion, Größer, Gleich usw.) auf Felder angewendet werden. Es kann gruppiert und sortiert werden. In diesen Rahmen werden aber nur einfache `Selects` behandelt.

*Alle Daten der Tabelle **tierfamilie** ausgeben:*

```
SELECT * FROM tierfamilie;
```

Ergebnis:

```
id tierfamilie
 1 Säugetiere
 2 Fische
 3 Reptilien
 4 Vögel
```

*Nur das Feld **tierfamilie** der Tabelle **tierart** alphabetisch sortiert ausgeben:*

```
SELECT tierart FROM tierart ORDER BY
tierart;
```

Ergebnis:

```
tierart
Goldfisch
Hunde
Katzen
Nagetiere
```

*Die Tabellen **tierfamilie** und **tierart** verknüpfen und alle Datensätze ausgeben:*

```
SELECT tierfamilie, tierart
FROM   tierfamilie, tierart
WHERE  tierfamilie.id =
tierart.tierfamilie_id;
```

Ergebnis:

```
tierfamilie  tierart
Säugetiere   Hunde
Säugetiere   Katzen
Säugetiere   Nagetiere
Fische       Goldfisch
```

Ausgeben der Anzahl der Tierarten für jede Tierfamilie:

```
SELECT tierfamilie, COUNT(*) AS anzahl
FROM   tierfamilie, tierart
WHERE  tierfamilie.id =
tierart.tierfamilie_id
GROUP BY tierfamilie;
```

Ergebnis:

```
tierfamilie anzahl
Fische           1
Säugetiere       3
```

Beim letzten Beispiel wurde eine `GROUP BY`-Klausel verwendet, damit die Tierart nur einmal ausgegeben wird und der Zählvorgang durchgeführt werden kann. `COUNT(*)` heißt nur, dass die Datensätze gezählt werden sollen. "`AS anzahl`" bedeutet, dass die Zahl, welche `COUNT(*)` zurück liefert den Namen "`anzahl`" bekommt. Wie man sieht, werden die Tierfamilien nicht angezeigt, welche keine Tierart zugeordnet haben. Dieses Problem löst man mit einem `LEFT JOIN` wie im folgenden Beispiel:

```
SELECT tierfamilie, COUNT(tierart.id) AS
anzahl
FROM   tierfamilie LEFT JOIN tierart
  ON
   tierfamilie.id =  tierart.tierfamilie_id
GROUP BY tierfamilie
ORDER BY anzahl DESC;
```

Ergebnis:

```
tierfamilie anzahl
Säugetiere       3
Fische           1
Reptilien        0
Vögel            0
```

`LEFT JOIN` liefert jetzt auch Tierfamilien zurück, welche in der Tabelle Tierart keinen zugehörigen Datensatz haben. Im `COUNT`-Befehl muss jetzt ein Feld der Tabelle **`Tierart`** angegeben werden, in diesem Fall die **`Tierart-Id`**. Die **`ORDER BY`**-Klausel sagt aus, dass nach der ermittelten Anzahl absteigend (`DESC`) sortiert werden soll.

2.4.2 INSERT-Statement

```
INSERT INTO table_name [ (column_name,...) ]
VALUES (expression,...)
```

Das `INSERT`-Statement dient zum Erzeugen neuer Datensätze in einer Tabelle. Die Liste der Feldnamen kann weggelassen werden, wenn in alle Felder etwas eingefügt werden soll. In diesem Fall muss man aber die Werte in genau der Reihenfolge angeben, wie sie beim `CREATE TABLE` definiert wurden. Bei den Werten müssen Zeichenketten und Datum in einfachen Hochkomma stehen, nur für Zahlen gilt das nicht.

Einfügen einer neuen Tierart "Delphin":

```
INSERT INTO tierart (id, tierfamilie_id,
tierart) VALUES (15,1,'Delphin');
```

2.4.3 UPDATE-Statement

```
UPDATE table_name SET column=expression,...
[WHERE conditions]
```

Das `UPDATE`-Statement dient zum Ändern eines vorhandenen Datensatzes. Dabei können beliebige Felder geändert werden, diese müssen nur nacheinander durch Komma getrennt nach dem Schlüsselwort `SET` angegeben werden. Die `WHERE`-Bedingung sagt aus, welcher oder welche Datensätze geändert werden sollen. Gibt man keine `WHERE`-Bedingung an, werden alle Datensätze in der angegebenen Tabelle geändert!

Ändern der Tierart "Delphin" in "Delphine"

```
UPDATE tierart SET tierart='Delphine' WHERE
id=15;
```

2.4.4 DELETE-Statement

```
DELETE FROM table_name [WHERE conditions]
```

Mit dem `DELETE`-Statement kann man einen oder mehrere Datensätze aus einer Tabelle löschen. Die `WHERE`-Bedingung dient zum Einschränken der zu löschenden Datensätze. Wird keine `WHERE`-Bedingung angegeben, werden alle Daten ohne Nachfrage gelöscht!

Löschen der Tierart "Delphine"

```
DELETE FROM tierart WHERE id=15;
```

3 Datenbankanbindung mit mySQL und PHP

3.1 Was ist mySQL?

MySQL ist ein echtes Multi-User und Multi-Threaded relationales Datenbanksystem, das mit Client/Server-Architektur arbeitet. Es ist plattformunabhängig und wird für Unix-Rechner im nichtkommerziellen Bereich als Freeware angeboten. Client/Server-Architektur bedeutet, dass mySQL auf dem Server im Hintergrund läuft, aber nur einmal geladen werden

SELECT	{*|column_name [, column_name]}	→ **Feld(er), * alle Felder**
FROM	table_name [, table_name]	→ **Tabelle(n)**
WHERE	conditions	→ **Bedingung(en)**
ORDER BY	column_name [, column_name]	→ **Sortierung**

http://www.tierschutz.at/

muss. Dieser Dämon (Hintergrundprozess) nimmt alle Datenbankzugriffe entgegen und leitet die Ergebnisse an die Clients weiter. Die wichtigsten Eigenschaften von mySQL sind Geschwindigkeit, Stabilität und einfache Bedienbarkeit.

3.2 Was ist PHP?

PHP steht für "*PHP Hypertext Preprocessor*" und ist eine serverseitige Skript-Sprache zur Erstellung von dynamischen Webseiten. Die Anweisungen der Sprache sind dabei in den HTML-Code einer Webseite eingebettet. Der PHP-Code wird auf dem Server verarbeitet, daher bekommt der Benutzer beim Aufruf einer PHP-Seite nichts vom Code zu sehen. Die Syntax von PHP ist ähnlich wie die von C oder JavaScript. PHP ist relativ einfach zu erlernen, unterstützt ein Vielzahl von Datenbanksystemen und kann durch eine Vielzahl von Modulen um wichtige Funktionen erweitert werden. Die erste Version von PHP wurde 1994 von Rasmus Lerdorf aus Kanada entwickelt. Heute kümmern sich mehrere Entwickler auf der ganzen Welt um PHP.

3.3 Aufgabenstellung für unser Beispiel

Wir werden ein kleines Redaktionssystem entwickeln, welches Artikel verwalten kann. Ein Artikel besteht aus Überschrift und Text.

Datenbank und Tabelle mit mySQL anlegen

Das Datenbanksystem mySQL kann über mehrere Schnittstellen angesprochen werden. Hier wird die Kommandozeile verwendet, in der hauptsächlich die Administration durchgeführt wird. Wir werden jetzt die Datenbank und deren Tabellen anlegen:

Datenbank redsys anlegen

```
mysql> CREATE DATABASE redsys;
```

Dem System mitteilen, dass sich Anweisungen auf die Datenbank ***redsys*** *beziehen:*

```
mysql> USE redsys;
```

Die Tabelle ***artikel*** *anlegen*

```
mysql>
CREATE TABLE artikel (artikel_id
   INT not null AUTO_INCREMENT,
   titel VARCHAR (255) not null,
   text TEXT,
   PRIMARY KEY (artikel_id),
   INDEX (artikel_id),
   UNIQUE (artikel_id));
```

`AUTO_INCREMENT` sagt aus, dass mySQL dieses Feld von selbst füllen soll und zwar immer um eins erhöht gegenüber dem höchsten Wert in diesem Feld.

Wir fügen gleich den ersten Datensatz ein:

```
mysql>
INSERT INTO artikel (titel, text)
VALUES ('Mein erster Artikel',
        'Jetzt kommt der Inhalt');
```

3.4 Verbindung zur Datenbank mittels PHP herstellen

In unserem PHP-Script müssen wir zuerst eine Verbindung mit der Datenbank herstellen. Dafür gibt es den Befehl `mysql_connect`. Als Parameter muss der Hostname angegeben werden. Weiters kann man Benutzername und Passwort angeben. Im Fehlerfall wird die Anweisung "`die`" ausgeführt (das Skript wird beendet) und eine Fehlermeldung wird ausgegeben:

```
mysql_connect("localhost","user","passwd")
or die("Unable to connect to SQL server");
```

Als Nächstes muss auch hier wieder die Datenbank ausgewählt werden. Dies geschieht mit dem Befehl `mysql_select_db`:

```
mysql_select_db("redsys")
or die("Unable to select database");
```

Um beispielsweise ein `SELECT`-Statement abzusetzen verwendet man den Befehl `mysql_query`:

```
$artikel = mysql_query("SELECT * FROM
artikel") or die("Select failed!");
```

3.5 PHP-Scripts zum Anzeigen, Einfügen, Ändern und Löschen

3.5.1 Anzeigen aller Artikel

Als ersten Schritt werden wir ein PHP-Script schreiben, welches alle Artikel aus der Datenbank liest und diese sortiert nach dem Schlüsselfeld ausgibt. Zunächst benötigen wir wieder den Datenbankverbindungsaufbau. Danach setzten wir ein `Select` ab, welches alle Artikel ausliest. Das Ergebnis müssen wir mit einer `while`-Schleife und dem Befehl `mysql_fetch_array` auslesen und ausgeben. Der Befehl `mysql_fetch_array` speichert die Daten in einem assoziativen Array, d.h. die Indizes entsprechen dem Namen der Datenbankfelder.

```
<html>
<head>
   <title>Artikel anzeigen</title>
</head>
<body>
<H2>Artikel anzeigen</H2>
<a href="artikel_einfuegen.htm">Artikel
hinzuf&uuml;gen</a>
<br>
<?
mysql_connect ("localhost") or die("Unable
to connect to SQL server");
mysql_select_db("redsys") or die("Unable to
select database");
$artikel = mysql_query("SELECT * FROM
artikel ORDER BY artikel_id DESC")
           or die("Select failed!");

while ($rec_artikel =
mysql_fetch_array($artikel))
{
   echo "<H3>" .
htmlentities($rec_artikel["titel"]) .
"</H3>";
   echo "<p>" .
htmlentities($rec_artikel["text"]) . "</p>";
   echo "<a href=\"artikel_aendern.php?id="
. $rec_artikel["artikel_id"];
   echo "\">Artikel &auml;ndern</a>
  ";
   echo "<a href=\"artikel_loeschen.php?id="
. $rec_artikel["artikel_id"];
   echo "\">Artikel l&ouml;schen</a>";
}
?>
</body>
</html>
```

Weiters geben wir noch drei Links aus: zum Hinzufügen, Ändern und Löschen eines Artikels. Beim Ändern und Löschen müssen wir allerdings dem nachfolgendem Skript die **`Artikel-Id`** mitgeben, damit das Skript weiß, um welchen Datensatz es sich handelt. Deshalb wird der Dateiname um "`?id=wert`" erweitert. Statt `wert` setzten wir die **`Artikel-Id`** aus dem Array des gelesenen Datensatz ein.

Die Funktion `htmlentities` wandelt den Text in HTML-Code um (z.B. Umlaute). Das Ergebnis sieht im Browser so wie in der oberen Abbildung aus. Der erste Artikel den wir eingefügt haben, wird bereits angezeigt.

3.5.2 Einfügen eines Artikels

Um weitere Artikel ohne Kenntnisse von SQL einfügen zu können, benötigen wir erstens ein Formular, in welchem die Daten eingegeben werden können, und zweitens ein PHP-Script, welches die im Formular enthaltenen Daten in die Datenbank überträgt. Das erste Dokument enthält nur HTML-Code mit dem Formular und sieht folgendermaßen aus (die BODY- und HTML-Tags wurden weggelassen):

```
<form method="post"
action="artikel_insert.php">
   Titel: <input type="text" name="titel"
size=60 maxlength=255><br><br>
   Text:  <textarea name="text" cols=45
rows=4></textarea><br><br>
   <input type=submit value="einf&uuml;gen">
</form>
```

Im zweiten Dokument befindet sich nur PHP-Code. Das Script muss jetzt einerseits eine Datenbankverbindung herstellen und andererseits das INSERT-Statement mit den richtigen Daten aus dem Formular füllen und absetzen. Die eingegebenen Daten sind in den Variablen `$titel` und `$text` gespeichert. Diese Variablen heißen so wie sie in den Formular-Tags im Feld `name=` angegeben wurden. Nach dem Einfügen soll wieder auf die "Artikel anzeigen"-Seite umgeleitet werden. Das geschieht mit dem Befehl "`header`".

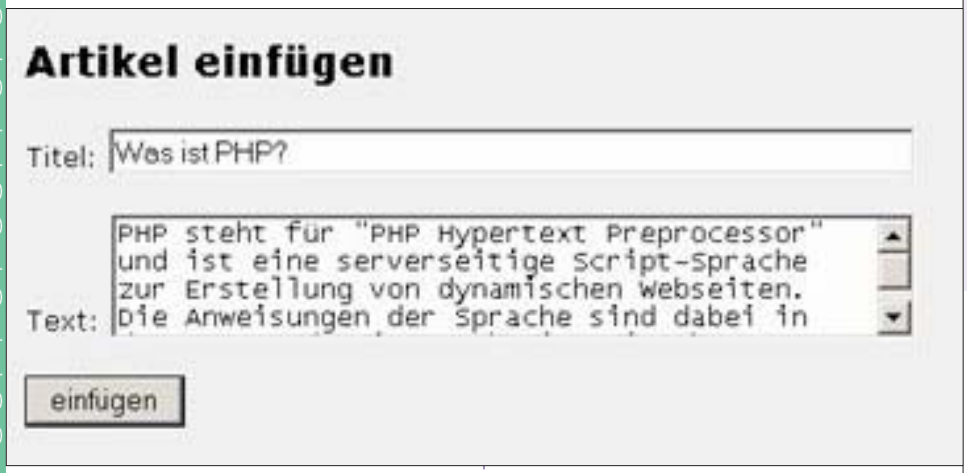
```
<?
mysql_connect ("localhost") or die("Unable
to connect to SQL server");
mysql_select_db("redsys") or die("Unable to
select database");

$query = "INSERT INTO artikel (titel, text)
";
$query.= "VALUES ('" . $titel . "','" .
$text . "')";
mysql_query($query) or die("Insert
failed!");

header("Location: artikel_anzeigen.php");
?>
```

3.5.3 Ändern eines Artikels

Um nachträglich auch Artikel ändern zu können, brauchen wir wieder zwei Skripts. Das erste ist diesmal kein reines HTML-Dokument, sondern muss mittels PHP die Daten die geändert werden sol-

http://www.tierschutz.at/

Und so sieht das ausgefüllte Formular im Browser aus:



Der neue Artikel ist an die erste Stelle gerückt:

len in das Formular laden. Dazu wird mit einem `Select` der Datensatz gelesen. In diesem `Select` wird die übergebene **`Artikel-Id`** benötigt, damit die richtigen Daten in das Formular kopiert werden. Das Skript sieht so aus:

```
<?
mysql_connect ("localhost") or die("Unable
to connect to SQL server");
mysql_select_db("redsys") or die("Unable to
select database");

$query = "SELECT * FROM artikel WHERE
artikel_id=" . $id;
$artikel = mysql_query($query) or
die("Select failed!");
$rec_artikel = mysql_fetch_array($artikel);

echo "<form method=\"post\"
action=\"artikel_update.php?id=";
echo "$rec_artikel["artikel_id"] . "\">";

echo "Titel: <input type=\"text\"
name=\"titel\" size=60 maxlength=255 ";
echo "value=\"" . $rec_artikel["titel"] .
"\"><br><br>";

echo "Text:  <textarea name=\"text\" cols=45
rows=4>";
echo "$rec_artikel["text"] .
"</textarea><br><br>";
echo "<input type=submit
value=\"einf&uuml;gen\">";
echo "</form>";
?>
```

Im zweiten Skript wird das Update durchgeführt und wieder auf die Anzeigen-Seite weitergeleitet:

```
<?
mysql_connect ("localhost") or die("Unable
to connect to SQL server");
mysql_select_db("redsys") or die("Unable to
select database");

$query = "UPDATE artikel SET titel='" .
$titel;
$query.= "', text='" . $text . "' WHERE
artikel_id = " . $id;
mysql_query($query) or die("Update
failed!");

header("Location: artikel_anzeigen.php");
?>
```

Im Browser sieht es dann so aus: ***Bild "Artikel ändern"***

Der Text des ersten Artikels wurde mit einem neuen Text überschrieben. In der nächsten Abbildung sehen wir die Änderung:

3.5.4 Löschen eines Artikels

Damit Artikel auch wieder gelöscht werden können, benötigen wir ein Skript, welches das `DELETE`-Statement absetzt. Und das sieht so aus:

```
<?
mysql_connect ("localhost") or die("Unable
to connect to SQL server");
mysql_select_db("redsys") or die("Unable to
select database");

$query = "DELETE FROM artikel WHERE
artikel_id = " . $id;
mysql_query($query) or die("Delete
failed!");

header("Location: artikel_anzeigen.php");
?>
```

http://www.tierschutz.at/