PHP und MySQLi

Der Benutzer kann mit Hilfe von PHP eine komfortable Schnittstelle zu den MySQL-Datenbanken herstellen, um Daten anzuzeigen, einzufügen, zu verändern und zu löschen.
Die Datenbank und der Strukturentwurf wurden vorher mit phpMyAdmin erzeugt.

Zur Zusammenarbeit zwischen PHP und MySQL stehen zwei Erweiterungen zur Verfügung, die MySQL-API und PDO.

- **MySQLi** i steht für MySQL Improved Extension und ist die neuere Entwicklung. Sie erlaubt zwei Arten der Programmierung: prozedual und objektorientiert. Außerdem bietet sie attraktive Features wie Prepared Statements. Die prozedurale Variante ist etwas sperrig und benötigt mehr Tipparbeit. Daher verwenden wir die objektorientierte Variante, die obendrein auch schneller arbeitet.
- **PDO** steht für „PHP Data Objects“ und ist nicht nur für MySQL ausgelegt. Diese Schnittstelle ist zwar leistungsfähiger, aber dafür auch etwas komplizierter als mysqli.

Erklärung:
Die Funktion (es gibt zwei Varianten)
- „new MySQLi()“ bzw. auch
- „mysqli_connect()“öffnet eine Verbindung zum MySQL- Datenbankserver.
  Der Rückgabewert der Funktion ist eine Referenz auf die Verbindung. Diese Referenz wird anschließend für weitere Funktionen benötigt und daher in der Variablen „$con“ gespeichert

Details:
new mysqli() erstellt ein neues Objekt. In der Klammer übergibt man zuerst den Namen des

- **Datenbankservers**, dann den
- **Benutzernamen für die MySQL-** Namen der **Datenbank (z.B. root)**, das
- **Passwort** und den
- **Datenbank**, auf die man zugreifen möchte. In unserem Fall hier (XAMPP) ist der Benutzer immer „root“ und kein Passwort vergeben.

$con dies ist eine Variable, die aber auch anders heißen könnte, wie z.B. $db oder $connection. Hauptsache, man kann sie später als Referenz zu Abfragen (query) verwenden, da man zwei Parameter bei einer Abfrage benötigt.

Besser inkl. error Meldung:

```
<?php
$con = mysqli_connect(“server_name”,”user_name”,”password”,”db_name”);
if (mysqli_connect_errno())    {
 echo “connection was not established ” . mysqli_connect_error();
}
?>
```

Weitere Infos:
http://www.php-kurs.com/mysql-datenbank-verbindung-herstellen.htm
https://www.youtube.com/watch?v=HQOS3V9nyEQ

MySQLi_Klasse:

- error liefert, sofern vorhanden, die Fehlermeldung des letzten MySQL-Befehls
- errno beinhaltet den nummerischen Fehlercode des letzten MySQL-Befehls (errno = error number)

MySQLi verwenden

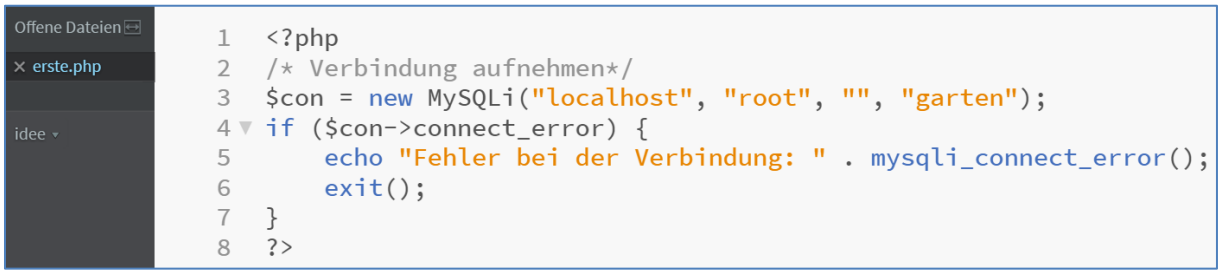
Mehrere Schritte sind nötig, um per PHP das Ergebnis einer MySQL-Abfrage auszugeben:

1. man muss eine Verbindung zum Datenbankserver erstellen
2. Dann muss man die Datenbank auswählen, auf die man zugreifen möchte. Dies kann auch gleichzeitig mit Schritt 1 erfolgen (passiert in MySQLi auch gemeinsam)
3. Dann muss man die Abfrage an die Datenbank senden
4. Als Nächstes muss das Ergebnis abgearbeitet und für die Webseite aufbereitet werden
5. Am Ende schließt man die Verbindung

0)Erstelle eine PHP-Site – speichern unter „1_db_verbindung.php

1+2)Verbindung erstellen und Datenbank auswählen

Die benötigten Angaben zur Kontaktaufnahme mit dem Datenbankserver eines externen Webservers bekommt man vom Provider z.B. www.bplaced.net. Bei einer lokalen Konfiguration mit XAMPP sind es der Hostname „localhost“ und der Username „root“. Ein Passwort ist nicht vergeben.




```php
<?php
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "garten");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}
?>
```

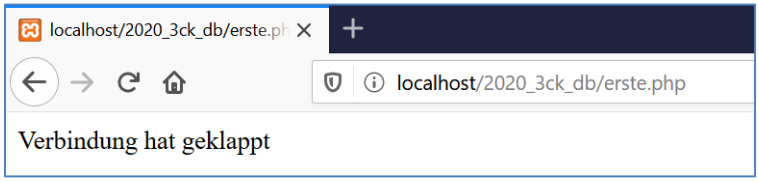
Code:

```
<?php
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "garten");
if ($con->connect_error) {e
cho "Fehler bei der Verbindung: " . mysqli_connect_error();
        exit();
}
echo "Verbindung hat geklappt";
$con->close();
?>
```

Man erhält ein Objekt zurückgeliefert, das die Verbindung zur Datenbank darstellt. Aber auch, wenn die Verbindung nicht funktioniert hat.

Daher wird zur Überprüfung auf die Eigenschaft „connect_error“ des mysqli-Objekts zurückgegriffen. Dieses beinhaltet eine Beschreibung des letzten Verbindungsfehlers. Ist ein Fehler aufgetreten, wird dieser ausgegeben und das Skript beendet mit exit(). Ansonsten erscheint die Meldung „Verbindung hat geklappt“ und die Verbindung wird über die close()-Methode geschlossen.

Ergebnis im Browser:



Für den nächsten Punkt 3) lösche diesen Teil des Codes, der „Verbindung hat geklappt“ ausgibt und dann das Skript schließt. Es soll ja hier noch nicht geschlossen werden!!!

```
 8
 9  echo "Verbindung hat geklappt";
10  ?>
```

Der Kontakt zum Datenbankserver ist also schon einmal geschafft.

Probe:

Probiere gleich einmal die Verbindung aus und lasse als Test die Art der Verbindung zum Server und seine Versionsnummer ausgeben.

Dazu dienen die beiden Eigenschaften
host_info und server_info
auf die man mit
$con ->host_info bzw.
$con ->server_info
zugreift.

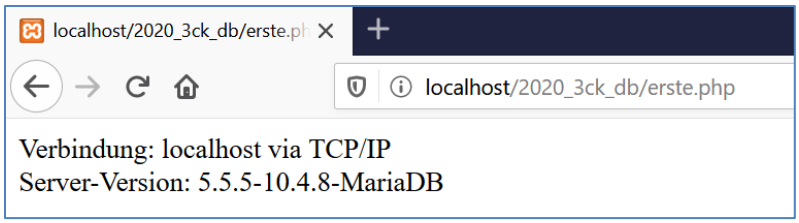
Die Ausgabe erfolgt mittels „echo“:

<?php
/* Probe*/
$con = new MySQLi("localhost", "root", "");

echo "Verbindung: ",$con->host_info, "
";

echo "Server-Version: ",$con->server_info, "
";

```
 8
 9  echo "Verbindung: ",$con->host_info, "<br>";
10  echo "Server-Version: ",$con->server_info, "<br>";
11
12  ?>
```



3)Abfrage durchführen und einfache Ausgabe

Es gibt 2 Schreibweisen:

1. $erg = $con->query("SELECT * FROM pflanzen");

```
5  $erg = $con->query("SELECT * FROM pflanzen");
6
```

Also eine neue Variable und nach dem Istgleichzeichen die bezugnehmende Variable und dann den Pfeil und den Begriff „query()“

Man kann aber auch gleichbedeutend schreiben:

2. $erg = mysqli_query($con, "SELECT * FROM pflanzen");

```
5  $erg = mysqli_query($con, "SELECT * FROM pflanzen");
6
```

Wenn die Verbindung steht, kann man Abfragen durchführen. Zur Übermittlung einer Anfrage an den Server steht, wie oben gerade gezeigt, auch die Methode „query()“ zur Verfügung. Mit einer „query()-Methode“ kann man in Klammern den SQL-Code direkt in Gänsefüßchen eingeben:
$erg = $con->query("SELECT * FROM pflanzen");

Man erhält ein Objekt vom Typ „mysqli_result“, das hier in der Variablen $ergebnis gespeichert wird. Auf das kann man nun mit verschiedenen Methoden zugreifen.
Der MySQL-Server liefert das Ergebnis immer in Form einer Tabelle aus. Bei $ergebnis handelt es sich nämlich um eine sogenannte Ergebnisliste, um ein „result set“. Die einzelnen Zeilen werden in Listenform in dieser Variablen gespeichert. Die Zeilen stehen im Prinzip untereinander, wie in einer Tabelle. Diese Ergebnisliste wird später Zeile für Zeile ausgelesen werden.

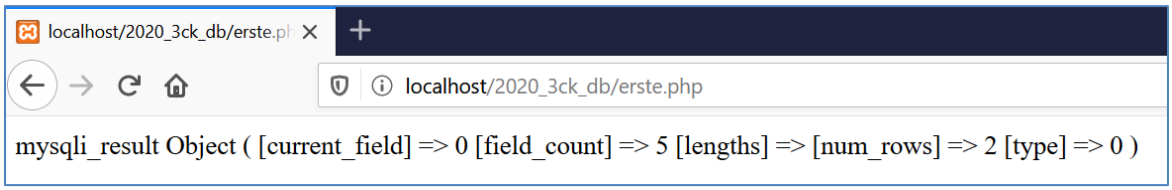
Das $con kommt aus der „Verbindungsaufnahme“ von vorhin und muss weiterverwendet werden.

Mit dem Sternchen * werden alle Daten der Tabelle ausgegeben.

```php
<?php
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "garten");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}

$erg = $con->query("SELECT * FROM pflanzen");
print_r($erg);
```

Zur schnellen Ausgabe reicht ein „print_r“:

Ergebnis im Browser:



4)Ergebnis für die schönere Ausgabe aufbereiten

Mit der Methode „fetch_array()“ holt man sich den ersten Datensatz des Ergebnisses in Form eines Arrays.

$zeile = $erg->fetch_array();

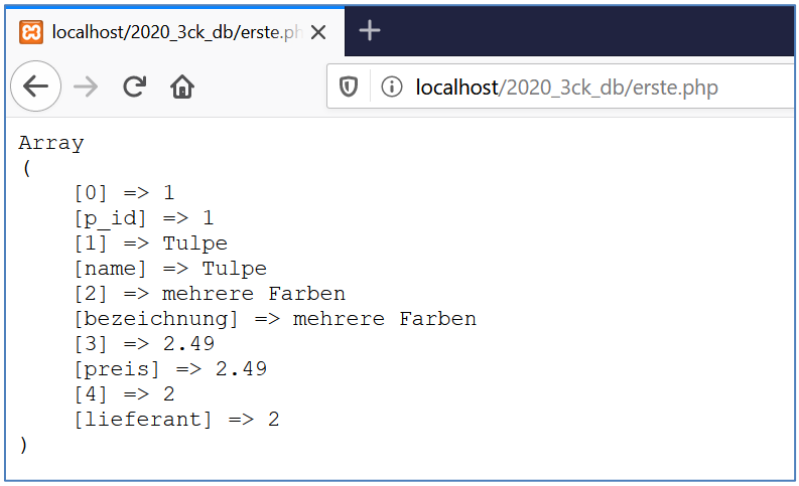
```php
$zeile = $erg->fetch_array();
```

Dieses Array kann für einen ersten Überblick mit „print_r()“ ausgegeben werden. Damit die Ausgabe von „print_r()“ auch im Browser übersichtlich ist, wird das HTML-Element „pre“ eingesetzt. Danach wird die Methode „close()“ für das „mysqli-result-Objekt“ aufgerufen und damit der Speicherplatz freigegeben, den das Objekt belegt hat.

```
echo "<pre>";
print_r($zeile);
echo "</pre>";
$ergebnis->close();
```

```php
$erg = $con->query("SELECT * FROM pflanzen");

$zeile = $erg->fetch_array();

echo "<pre>";
print_r($zeile);
echo "</pre>";
```

Ergebnis im Browser:



5)Verbindung schließen

Am Schluss sollte man die Verbindung wieder schließen.

$con->close();

Ergänzung für utf8:

Ergänzend wurde außerdem

set_charset("utf8")

geschrieben, das sicherstellt, dass als Zeichensatz auf dem Client wirklich UTF-8 eingesetzt wird. Somit werden die Umlaute auch korrekt dargestellt.

```
 8 ▼ if (!$con->set_charset("utf8")) {
 9       echo "Fehler beim Laden von UTF8" . $con->error;
10   }
11
```

Code:

```
if (!$con->set_charset("utf8")) {
   echo "Fehler beim Laden von UTF8" . $con->error;
}
```

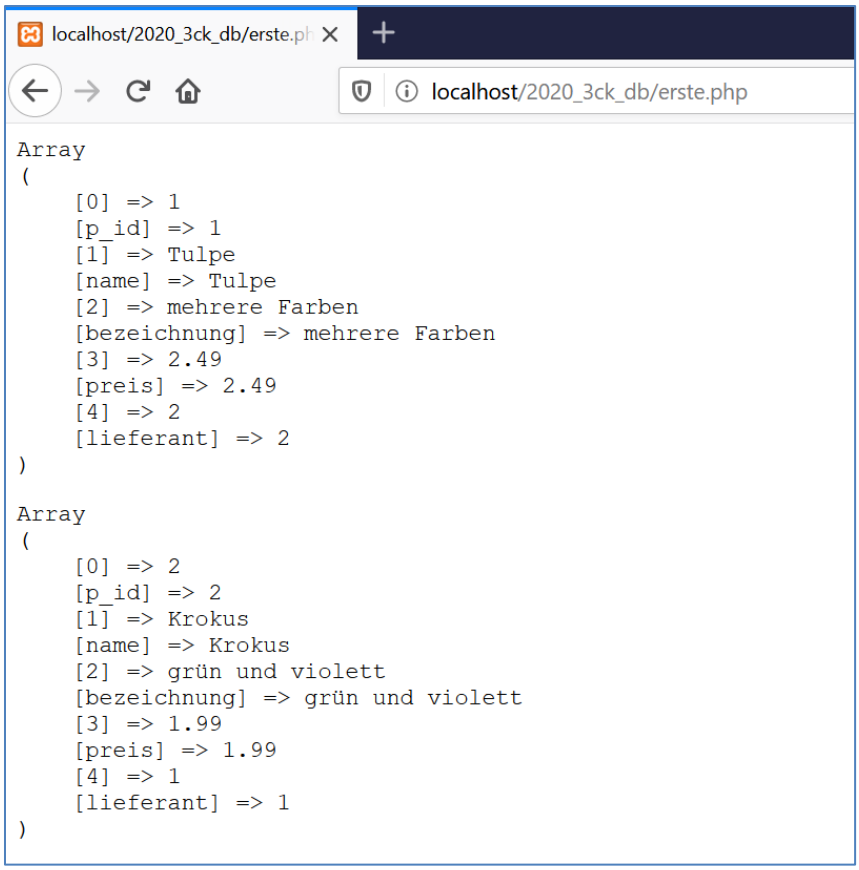
```php
<?php
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "garten");
if ($con->connect_error) {
    echo "Fehler bei der Verbindung: " . mysqli_connect_error();
    exit();
}
if (!$con->set_charset("utf8")) {
    echo "Fehler beim Laden von UTF8" . $con->error;
}

$erg = $con->query("SELECT * FROM pflanzen");

$zeile = $erg->fetch_array();

echo "<pre>";
print_r($zeile);
echo "</pre>";
```

Ergebnis besser aufbereiten

Die Methode „fetch_array()“ hat den Inhalt des ersten Datensatzes geliefert. Das Array enthält als Schlüssel einerseits Zahlen und andererseits die Spaltennamen. Das heißt, man kann den Preis der ersten Pflanze sowohl über $zeile [3] als auch über die $zeile [preis] sehen.
Aber „fetch_array()“ hat nur den ersten Datensatz geliefert. Um weitere Daten zu erhalten müsste man im Code mehrmalig das „fetch_array()“ einfügen (jedoch ohne „close“ zwischendurch):

```php
$erg = $con->query("SELECT * FROM pflanzen");

$zeile = $erg->fetch_array();

echo "<pre>";
print_r($zeile);
echo "</pre>";

$zeile = $erg->fetch_array();

echo "<pre>";
print_r($zeile);
echo "</pre>";
```

Ergebnis bei zweimal:



Quellen:

Florence Maurice in PHP 5.6 und MySQL 5.7, dpunkt Verlag, 2015, S.357-376

Giesbert Damaschke in PHP & MySQL, Wiley-Vch Verlag, 2015, S.281-288 und S.319-321