# L2\_1.3 Daten aus einer Datenbank auslesen und auswerten

In PHP existieren drei Varianten, um auf eine MySQL Datenbank zuzugreifen. Die älteste nutzt die MySQL Erweiterung, die aber seit PHP 5.5.0 als veraltet markiert wurde und in PHP 7 komplett entfernt wurde. Die zweite Möglichkeit ist mittels der MySQL Improved Extension (MySQLi) und die letzte Möglichkeit ist mittels PHP Data Objects (PDO). PDO ist dabei das aktuelle Interface und bietet eine Abstraktionsschicht für den Datenzugriff, das bedeutet, dass Sie, egal welches DBS Sie benutzen, dieselben Funktionen verwenden können, um Abfragen zu erstellen und Daten zu lesen. Wir werden uns ausschließlich mit dem modernsten Zugriff mittels PDO beschäftigen.

Ein PHP-Programm zur Ansteuerung eines DBMS enthält immer wesentliche Grundbausteine:

1. Der Verbindungsaufbau
2. Das Senden einer Anfrage an das DBMS
3. Die Auswertung der Antwort des DBMS
4. Der Verbindungsabbau

Hierfür stehen uns zwei Klassen zur Verfügung – die PDO-Basisklasse und die Klasse PDOStatement -, die wir im weiteren Verlauf kennenlernen. Beim Arbeiten mit einem DBMS können zahlreiche Fehler auftreten, die wir in unserem Programm erkennen und möglicherweise den Benutzer informieren müssen. Auf eine solche Fehlererkennung wird in diesem Beispiel zunächst verzichtet.

**1. Verbindungsausfbau**

Eine Verbindung zwischen PHP und dem DBMS ist Voraussetzung dafür, dass Sie SQL-Abfragen jeglicher Form an die Datenbank abschicken können und die entsprechenden Ergebnisse empfangen können. Um eine Verbindung herzustellen, erzeugen Sie zuerst ein Objekt der Klasse ***PDO***. Der Konstruktor erwartet drei Parameter zur Angabe der Datenbankquelle, für den Benutzername und das Passwort. Die Datenbankquelle wird dabei als Zeichenkette kodiert und durch den Typ des DBMS, den Namen des Rechners, auf welchem das DBMS läuft und den Namen der Datenbank bestimmt.

Bei Problemen mit der Darstellung von Umlauten zusätzlich einfügen

;charset=utf8

$pdo = new PDO("mysql:host=localhost;dbname=fahrschule", "root", "");

Datenbankquelle,mit den Bestandteilen „DBMS-Typ“, „Rechnername“, „Name der Datenbank“

Benutzername

Kennwort

In der Regel sieht der Verbindungsaufbau genau so aus, wie oben dargestellt. Lediglich der Name der Datenbank (roter Bereich) müssen wir gelegentlich anpassen. In der Standardinstallation von XAMPP lautet der Benutzername „root“. Das Kennwort ist leer, was durch eine leere Zeichenkette angezeigt wird (zwei Hochkommata direkt hintereinander). Bei erfolgreicher Verbindung zur Datenbank wird eine Instanz der PDO-Klasse an Ihr Skript zurückgegeben, welche hier in der Variablen $pdo gespeichert wird.

**2. Senden einer Anfrage an das DBMS**

Unter der Voraussetzung, dass der Verbindungsaufbau zum DBMS fehlerfrei funktioniert hat, kann im nächsten Schritt eine Anfrage mittels SQL formuliert und an das DBMS gesendet werden.

    $sql = 'SELECT vorname, nachname FROM fahrschueler';

    $stm = $pdo->query($sql);

„Antwortobjekt“ vom Typ PDOStatement

Aufruf der Methode query des PDO-Objektes $pdo. Die SQL-Anweisung wird in Form eines Parameters dem Methodenaufruf mitgegeben.

Zuerst definieren wir unsere SQL-Anweisung zur Abfrage der Datenbank als Zeichenkette und speichern diese in einer Variablen (hier: $sql). Anschließend wird die Methode query(…) der im ersten Schritt erstellten PDO-Instanz aufgerufen und die SQL-Anweisung als Parameter übergeben. Die Methode query sendet die Anfrage an das DBMS und wartet automatisch auf die Antwort. Liegt diese vor, dann liefert der Methodenaufruf diese als ein Objekt des Typs PDOStatement zurück. Dieses „Antwortobjekt“ wird in diesem Beispiel in der Variablen $stm gespeichert. Vereinfacht könnte man sich vorstellen, dass das Ergebnis – die Ergebnistabelle - der Abfrage nun in der Variablen $stm hinterlegt ist.



**$stm = $pdo->query($sql)**

**3. Auswertung der Antwort des DBMS**

Das Ergebnis der Abfrage befindet sich nun im Objekt $stm vom Typ PDOStatement. Es gilt nun, die Daten mit Hilfe dieses Objektes zu ermitteln. In der Regel besteht das Ergebnis der Abfrage aus mehreren Datensätzen. Die Auswertung erfolgt nun Datensatz für Datensatz! Hierzu nutzen wir eine Schleife, die solange ausgeführt wird, solange noch Datensätze im Ergebnis vorhanden sind.

Dieser Ausdruck liefert einen Datensatz aus dem Ergebnis als PHP-Objekt

    while ($row = $stm->fetchObject()) {

        echo $row->vorname . " ";

        echo $row->nachname . "<br>";

    }

In der Bedingung der while-Schleife sind zwei Anweisungen kombiniert. Zum einen liefert der Aufruf der Methode fetchObject(…) des Objektes $stm immer den nächsten Datensatz aus der Ergebnistabelle in Form eines PHP-Objekts, zum anderen wird dieses PHP-Objekt gleich der Variablen $row zugewiesen. Die Methode fetchObject(…)liefert solange einen „Datensatz“, solange noch ein Datensatz in der Ergebnistabelle verfügbar ist. Ist kein weiterer Datensatz verfügbar, dann liefert die Methode den Wert false, was zum Abbruch der Schleife führt.



**$row = $stm->fetchObject()**

**$row = $stm->fetchObject()**

**$stm**

1. Schleifendurchlauf

2. Schleifendurchlauf

**$row = $stm->fetchObject()**

3. Schleifendurchlauf

…

**. . .**

Der aktuell ausgelesene Datensatz befindet sich also in Form eines PHP-Objektes in der Variablen $row. Das Objekt $row zeichnet sich dadurch aus, dass es Eigenschaften besitzt, die genau so lauten, wie die Namen der Spalten der Ergebnistabelle. In diesem Beispiel besitzt das Objekt $row also die Eigenschaften „vorname“ und „nachname“. Diese können wir nutzen, um an die konkreten Werte zu gelangen:

SELECT vorname, nachname FROM fahrschueler



n. Schleifendurchlauf

**$row = $stm->fetchObject()**

Mit

$row->vorname

können wir hier auf den Wert „**Andreas**“ zugreifen,

mit

$row->nachname

auf den Wert „**Abele**“.

Merke:

Die Eigenschaften des von der Methode fetchObject(…) gelieferten Objektes heißen immer so, wie die Spaltennamen der Ergebnistabelle!

Möchte man diese Werte ausgeben, kann, wie im Beispiel oben, einfach eine echo-Anweisung vorangestellt werden.

        echo $row->vorname . " ";

        echo $row->nachname . "<br>";

4**. Verbindungsabbau**

Die in Schritt 1 erstellte Verbindung bleibt während der Lebensdauer des PDO-Objekts aktiv. Um die Verbindung zu beenden, müssen Sie das Objekt löschen, indem Sie sicherstellen, dass alle Referenzen darauf gelöscht werden - das erreichen Sie, indem Sie der Variable, die das Objekt beinhaltet, den Wert NULL zuweisen. Wenn Sie das nicht explizit tun, schließt PHP die Verbindung automatisch, wenn Ihr Skript endet. Gleiches gilt für das Objekt des Typs PDOStatement.

$stm = null;

$pdo = null;

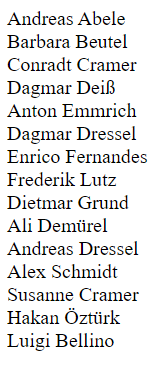
**Zusammenfassung**

Fügt man alle Schritte aneinander, so ergibt sich folgendes PHP-Skript.

 <?php

*//1. Verbindungsaufbau*

    $pdo = new PDO('mysql:host=localhost;dbname=fahrschule', 'root', '');

    *//2. Formulierung und Senden der Anfrage*

    $sql = 'SELECT vorname, nachname FROM fahrschueler';

    $stm = $pdo->query($sql);

*//3. Auswertung des Ergebnisses*

    while ($row = $stm->fetchObject()) {

        echo $row->vorname . " ";

        echo $row->nachname . "<br>";

    }

    $stm = null;

*//4. Verbindungsabbau*

    $pdo = null;

?>

Anmerkungen: Die in diesem Beispiel verwendeten Variablen können natürlich auch anders benannt werden. In der Praxis finden Sie sehr häufig die hier verwendete Namenswahl.