PHP: PDO-Module

Die PHP Data Objects-Erweiterung (PDO) stellt eine Schnittstelle für den Zugriff auf Datenbanken zur Verfügung. Die einzelnen Funktionen ermöglichen es, Abfragen zu erstellen, Daten zu lesen und zu manipulieren. Die Interaktion zeigt Abbildung 1:

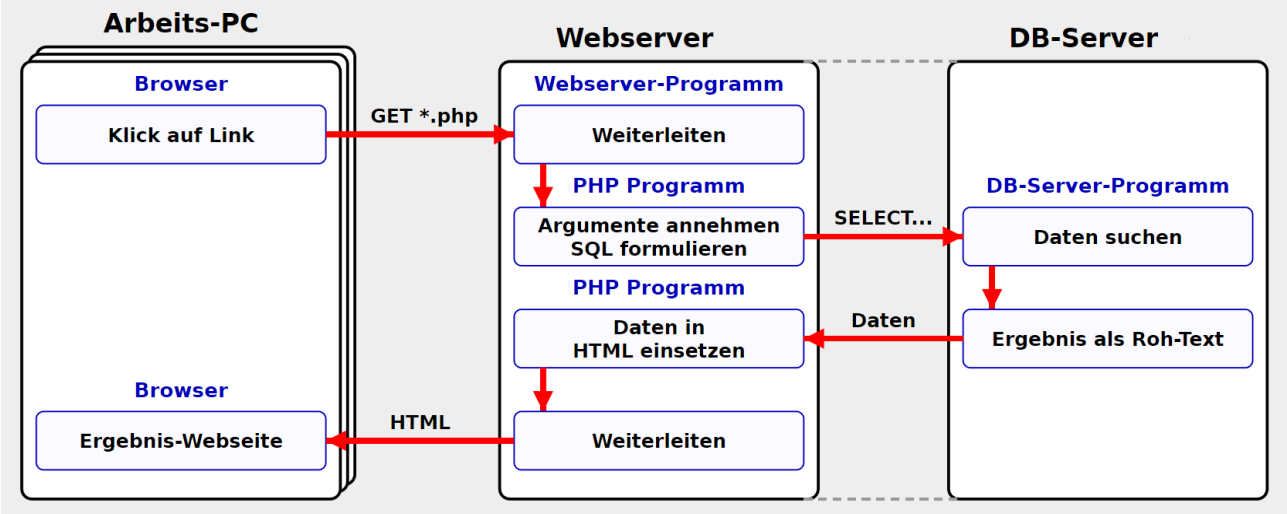


Abbildung 1: Client-Server-Interaktion

Nach dem Aufbau der Verbindung zwischen Webserver und Datenbankserver können SQL-Anweisungen an den DB-Server gesendet werden. Die daraus resultierenden Ergebnisse können in Variable bzw. in assoziativen Arrays übernommen werden. So können sie in den HTML-Code eingebaut und an den anfragenden Client gesendet werden.

Achtung: Die folgenden Codeteile enthalten keine Fehlerbehandlung oder Sicherheitsabfragen bezüglich der Gültigkeit der Eingaben!

1 Datenbankverbindung

In der Datei `dbcon.php` wird die Verbindung zur Datenbank hergestellt. Diese Datei wird mit `include_once('dbcon.php');` in der ersten Zeile jeder Datei eingebunden, in der ein Zugriff auf Daten dieser Datenbank benötigt wird.

Dazu sind folgende Daten erforderlich:

- die **Adresse des Servers**, meist `localhost` oder `127.0.0.1`, wenn der DB-Server und der Apache Webserver auf dem gleichen System laufen. Manchmal muss nach einem Doppelpunkt der Port angegeben werden, z. B. `localhost:3306` .
- die Bezeichnung bzw. der **Name der Datenbank**
- den **Benutzernamen** des Benutzers, der alle Rechte für die Datenbank besitzt
- das **Passwort des Benutzers**

1.1 Quellcode

```
<?php
$servername = "localhost";
$dbname = "datenbank";
$username = "root";
$password = "";
$conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
?>
```

Mit `new PDO` wird in der fünften Zeile eine Instanz der PDO-Basisklasse erzeugt, welche die Verbindung herstellt.

1.2 Grundstruktur einer PDO-SQL-Anfrage

In den letzten drei Zeilen der Datei `dbcon.php` wird die Codierung auf `utf8` eingestellt, damit alle Zeichen richtig dargestellt werden.

```
$statement = "SET NAMES utf8";
$sql = $conn->prepare($statement);
$sql->execute();
```

Sie zeigen die Grundstruktur einer Anweisung von PHP aus an den SQL-Server:

1. Bereitstellen eines SQL-Statements, das ist eine SQL-Abfrage bzw. SQL-Anweisung.
2. Vorbereiten des Statements durch den Datenbankserver liefert ein **Prepared Statement** für die Ausführung. Dies ist eine Art kompilierte Vorlage für die SQL-Anweisung.
3. Das **Prepared Statement** wird exekutiert (ausgeführt).

2 Datensatzgruppen – SELECT

Eine Datensatzgruppe (**Recordset**) ist das Ergebnis einer SELECT-Anweisung.

2.1 Ein Datensatz

Dateiname: **`tab_det.php`** für die Anzeige der Details eines Datensatzes.

Das Ergebnis umfasst nur eine Zeile:

```
$statement = "SELECT * FROM tabelle WHERE bedingung";
$sql = $conn->prepare($statement);
$sql->execute();
$recordset = $sql->fetch(PDO::FETCH_ASSOC);
```

Zeile 3 erzeugt eine Reihe von Objekten, über die in der Folge verfügt werden kann. In der letzten Zeile wird das Ergebnis der Abfrage in das assoziative Array `$recordset` übernommen (`fetch`). Dort kann auf die einzelnen Elemente über die Feldbezeichner (Spaltenüberschriften) aus der Datenbank zugegriffen werden. Zum Beispiel wird mit

```
<?php echo $recordset['tab_id']; ?>
```

der Inhalt des Feldes `tab_id` des ausgewählten Datensatzes in den HTML-Code geschrieben.

2.2 Mehrere Datensätze

Dateiname: **tab_lst.php** für die Anzeige einer Liste von Datensätzen.

Das Ergebnis umfasst mehrere Datensätze:

```
$statement = "SELECT * FROM tabelle WHERE bedingung ORDER BY feld";
$sql = $conn->prepare($statement);
$sql->execute();
$recordset_liste = $sql->fetchAll(PDO::FETCH_ASSOC);
$recordset_anzahl = $sql->rowCount();
```

In der vorletzten Zeile wird das Ergebnis der Abfrage in das assoziative Array `$recordset_liste` übernommen (`fetchAll`).

In der letzten Zeile erhält man die Anzahl der Zeilen, die sich aus der Abfrage ergeben haben, in der Variablen `$recordset_anzahl`.

Die Ausgabe erfolgt in einer `foreach` Schleife, zum Beispiel als Tabelle:

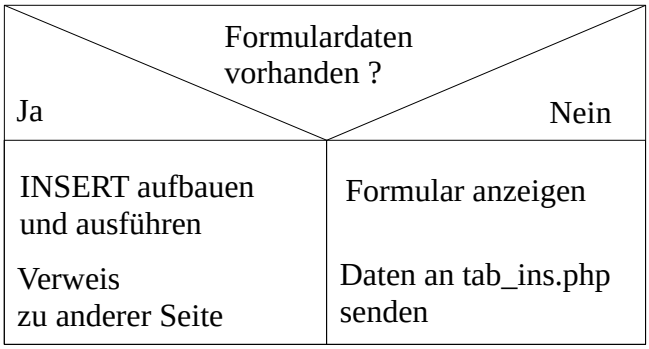
```
<table>
<?php foreach ($recordset_liste as $row => $recordset_link) { ?>
    <tr>
        <td><?php echo $recordset_link['tab_id']; ?></td>
        <td><?php echo $recordset_link['tab_name']; ?></td>
        <td><?php echo $recordset_link['tab_anschrift']; ?></td>
        <td><?php echo $recordset_link['tab_plz']; ?></td>
        <td><?php echo $recordset_link['tab_ort']; ?></td>
    </tr>
<?php } ?>
</table>
```

3 Datensatz einfügen – INSERT

Neue Daten werden in eine Datenbanktabelle eingefügt.

Dateiname: **tab_ins.php** für das Einfügen eines Datensatzes.

3.1 Struktur der Seite



3.2 Formular

Der neue Datensatz wird über ein Formular eingegeben. Jedes Formularfeld wird mit der Bezeichnung des entsprechendes DB-Felds benannt (`id` und `name`).

```
<form method="post" action="tabelle_ins.php" name="form1" id="form1">
    <p>Name: <input type="text" id="name" name="name" value=""></p>
    <p><button type="submit">Einfügen</button></p>
    <input type="hidden" id="insert" name="insert" value="form1">
</form>
```

Beim Absenden wird die Datei selbst aufgerufen und die Formulardaten übergeben. Das hidden-field `insert` dient zum Erkennen, ob das Formular ausgefüllt wurde. Am Beginn der Datei muss daher entschieden werden, ob Formulardaten vorhanden sind. Ist das nicht der Fall, so wird dieser Teil ignoriert und das Formular angezeigt.

3.3 Auswertung mit PHP

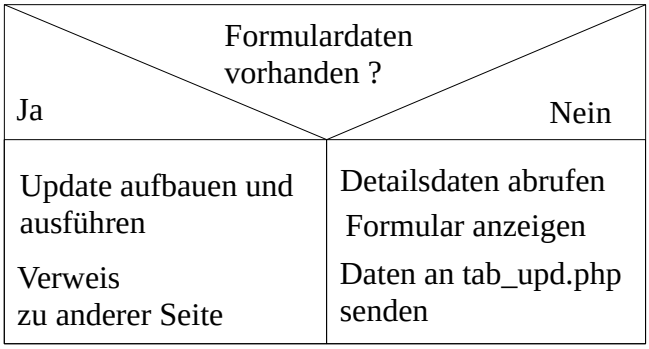
```
if ((isset($_POST["insert"])) && ($_POST["insert"] == "form1")) {
    //ist das Formular ausgefüllt, dann …
    //Formularparameter einlesen
    $name = $_POST["name"];
    //SQL - Anweisung
    $statement="INSERT INTO tabelle (name) VALUES ($name)";
    $sql = $conn->prepare($statement);
    $sql->execute();
    //Weiter zu einer anderen Seite
    header("Location: tabelle_lst.php");
}
```

4 Datensatz ändern – UPDATE

Ändern eines bestehenden Datensatzes.

Dateiname: **`tab_upd.php`** für das Einfügen eines Datensatzes.

4.1 Struktur



Der Aufbau ist ähnlich der Datei zum Einfügen von Datensätzen. Allerdings muss das Formular mit den Daten des zu ändernden Datensatzes ausgefüllt sein und es kommt die UPDATE-Anweisung zum Einsatz.

4.2 Formular

```
<form method="post" action="tabelle_upd.php" name="form1" id="form1">
    <p>Name: <input type="text" id="name" name="name"
                    value="<?php echo $tabelle['name']; ?>"></p>
    <p><button type="submit">Speichern</button></p>
    <input type="hidden" id="update" name="update" value="form1">
    <input type="hidden" id="tab_id" name="tab_id"
           value="<?php echo $tabelle['tab_id']; ?>">
</form>
```

4.3 Auswertung mit PHP

Am Beginn der Datei steht

```php
if ((isset($_POST["update"])) && ($_POST["update"] == "form1")) {
    //Formularparameter einlesen
    $tab_id = $_POST["tab_id"];
    $tab_name = $_POST["tab_name"];
    //SQL - Anweisung
    $statement = "UPDATE tabelle SET tab_name=$tab_name WHERE tab_id=$tab_id";
    $sql = $conn->prepare($statement);
    $sql->execute();
    // weiter zur Detailseite dieses Datensatzes
    header("Location: tab_det.php?tab_id=".$tab_id);
}

// Daten des ausgewählten Datensatzes einlesen (zum Ausfüllen des Formulars)
$tab_id = $_GET['tab_id'];
$statement = "SELECT * FROM tabelle WHERE tab_id=$tab_id";
$sql = $conn->prepare($statement);
$sql->execute();
$tabelle = $sql->fetch(PDO::FETCH_ASSOC);
```

4.4 Datensatz löschen

Vorsicht: Keine Sicherheitsabfragen!

Auf einer Seite wird ein Formular zum Löschen eines Datensatzes erstellt.

Dateiname: **`tab_ins.php`** für das Einfügen eines Datensatzes.

Die Auswahl erfolgt über die ID des Datensatzes. Zum Beispiel mit einem Formular:

```php
<form action="tab_del.php" method="post">
    <input type="hidden" id="tab_id" name="tab_id" value="<?php echo $tab_id; ?>">
    <button type="submit">Löschen</button>
</form>
```

Das Script zum Löschen steht in einer eigenen Datei `tab_del.php`, mit der nichts ausgegeben wird. Es wird nur der Datensatz gelöscht und dann in der letzten Zeile sofort zu einer anderen Datei `andereseite.php` weitergeleitet:

```php
<?php
include_once('dbcon.php');
//Löschen eines Datensatzes - keine Sicherheitsabfrage!
//POST-Parameter einlesen
$tab_id = $_POST["tab_id"];
//SQL - Anweisung
$statement = "DELETE FROM tabelle WHERE tab_id = $tab_id";
$sql = $conn->prepare($statement);
$sql->execute();
//weiter zu einer anderen Seite:
header("Location: andereseite.php");
?>
```