Probeprojekt: Adresslistenabfrage

Vertiefung Datenbank-Klasse

```
class mysqldb {

    /**
     * Connect to db, return handle
     *
     * @param string        $dbusername   DB username
     * @param string        $dbpassword   DB password
     * @param string        $dbname       Database to connect to
     * @param string        $host         Database host to connect to
(optional)
     *
     * @return object       DB-Handle
     */
    function mysqldb($dbusername, $dbpassword, $dbname, $host="localhost" )
{

    try {
        $pdo = new PDO("mysql:host=$host;dbname=$dbname", "$dbusername",
"$dbpassword");
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    } catch ( PDOException $e ) {
        echo 'Verbindung zur Datenbank fehlgeschlagen: ' . $e->getMessage();
        exit();
    }

    $this->connection = $pdo;

    }

    function getConnection() {
        return $this->connection;
    }

    function setQueryTable($table) {
        $this->qtable = $table;
    }

    function searchName($searchname) {
        $statement= $this->connection->prepare("SELECT * FROM $this->qtable
WHERE Nachname LIKE :nachname");
        //$statement->execute(array($table,$searchname));
        $statement->execute(array('nachname' => "%$searchname%"));
```

```
        // $statement->fetch() leifert das Ergebnis der Abfrage zeilenweise
        // https://www.php.net/manual/de/pdostatement.fetch.php
        // Für uns ist es meist besser ein assoziatives Array
        // mit den Ergebnissen zurückzugeben das dann gerendert werden
        // kann
        // Leeres Array definieren
        $resultArray = array();
        while($row = $statement->fetch(PDO::FETCH_ASSOC)) {
            // Die Zeilen an das Array anhängen (Push)
            $resultArray[] = $row;
        }

        // For Debugging
        // print_r($resultArray);

        // Ergebnisarray zurückgeben
        return $resultArray;
    }
```

Aufgaben

(A1)

Erweitere die bisherige Arbeit am Plugin auf ein "Adresslistenausgabewerkzeug":

(1) Lege die fiktive Adressdatenbank zugrunde, importiere die Adresstabelle in deine mysql-DB um sie dann abzufragen.

(2) Ergänze dein Plugin um eine Konfigurations-Option "databasetable", damit du in der Konfiguration festlegen kannst, welche Tabelle abgefragt werden soll.

(3) Erstelle ein – zunächst einfaches – Abfrageformular, mit dem du z.B. anhand des Vor- und/oder Nachnamens nach der Adresse einer Person suchen kannst. Implementiere die Abfrage als Methode in der mysqldb-Klasse, die das DB-Handle und die Abfrageparameter erhält und ein Array mit dem Ergebnis der Abfrage zurückgibt. Dieses Array kann die render Funktion dann als hübsche Tabelle ausgeben.

(4) Nachdem die Grundfunktionalität sichergestellt ist, kannst du das Abfrageformular und die Programmlogik um weitere Funktionen erweitern. (Sortierung, weitere Felder, optionale Suche nach Teilstrings in Datenbankfeldern u.v.m.)

(A2)

Erweitere dein Plugin um ein Formular, mit dem ein neuer Datensatz an die Tabelle angefügt werden kann.

(1) Welches der Formulare angezeigt werden soll, kann z.B. durch den Parameter nach dem > gesteuert werden:

```
{{projekt>abfrage}} // rendert und bearbeitet die Abfrage
{{projekt>eingabe}} // rendert und bearbeitet das Eingabeformular
```

(2) Beachte Sicherheitsaspekte und führe eine Eingabeüberprüfung durch, bevor du die Eingabedaten an deine Abfrage übergibst.