Neue Klassen mit Datenbankzugriff

Wir wollen unserem Projekt nun weitere Klassen hinzufügen, von denen einige Datenbankzugriff benötigen. Die jeweiligen Operationen auf der Datenbank sollen aber in den jeweiligen Klassen gekapselt sein.

Um dieses Ziel zu erreichen, *reduzieren* wir zunächst die Funktionalität unserer mysqldb-Klasse darauf, eine [1)] Datenbankverbindung herzustellen und ein entsprechendes PDO-Objekt, das diese Verbindung repräsentiert, zurückzuliefern, welches dann innerhalb der anderen Klassen für den Datenbankzugriff verwendet werden kann.

Die neue DB Klasse

Passe die Datenbank Klasse in der Datei `mysqldb.class.php` folgendermaßen an:

```
class mysqldb {

    static public $connection = null;
    private $dbobject;

    /**
     * Constructor - connect to db, set handle attribute
     *
     * @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)
     *
     */

    private function __construct($dbusername, $dbpassword, $dbname, $host )
{

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

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

    }
```

```
    public static function getConnection($dbusername, $dbpassword, $dbname,
$host ) {
        // Wenn das Objekt noch keine PDO-Instanz hat
        // wird eine erzeugt
        if(self::$connection === null) {
            $dbobject = new mysqldb($dbusername, $dbpassword, $dbname,
$host);
        }

     //Und das PDO-Objekt zurückgeben
     return self::$connection;
    }

    // Das Klonen dieser Instanz verhindern.
    private final function __clone () {}

}
```

Anmerkungen:

- Die Schreibweise mit dem Doppel-Doppelpunkt ermöglicht den Zugriff auf statische Methoden und Felder, ohne ein Objekt der Klasse zu besitzen. Das benutzen wir, um in weiteren Klassen ein Datenbank-Handle zu erhalten.
- Die Klasse selbst ist so konstruiert, dass sie nur eine Datenbankverbindung aufbaut. Wenn noch keine Datenbankverbindung existiert, wird ein neues myqldb-Objekt samt Datenbankhandle erzeugt, andernfalls lediglich das Handle zurückgegeben.
- Außerdem verhindern wir, dass die Instanz geklont werden kann.

Benutzer

Nun wenden wir uns der Benutzer-Klasse zu. Diese hat zunächst vor allem die Aufgabe, zu überprüfen, ob der angemeldete Dokuwiki Benutzer bereits einen Eintrag als Blog-Benutzer in der Datenbank hat. Wenn nicht muss der Benutzer angelegt werden.

Erstelle im Unterverzeichnis `class` deines Plugins eine Datei `bloguser.class.php` und definiere darin die Klasse bloguser:

```
<?php

class bloguser {

    protected $db;           // Das DB-Handle


    //Konstruktor
    public function __construct ($dbusername, $dbpassword, $dbname,
$host="localhost" )
    {
```

```
        // PDO Connection erzeugen/holen und als
        // Objektattribut "speichern". Damit werden DB Zugriffe möglich.
        $this->db = mysqldb::getConnection($dbusername, $dbpassword, $dbname,
$host);
    }

    public function checkBlogUser ($dwusername) {
        print "Überprüfe, ob der Dokuwiki-Benutzer " .
$_SERVER['REMOTE_USER'] . " schon in der Datenbank angelegt ist.";

    }

}

?>
```

Das ganze kann man dann etwa so in der `syntax.php` einbinden:

```
[...]
public function render($mode, Doku_Renderer $renderer, $data)
    {
        // Wir rendern nur HTML, sonst nix!
        if ($mode !== 'xhtml') {
            return false;
        }

        // Breche die Verarbeitung ab, wenn der Benutzer nicht angemeldet
ist.
        if ( ! isset($_SERVER['REMOTE_USER']) ) {
            $renderer->doc .= "Um das Blog zu benutzen, müssen Sie sich
anmelden";
            return true;
        }
        // get settings
        $dbusername = $this->getConf('dbusername');
        $dbname = $this->getConf('dbname');
        $dbhost = $this->getConf('dbhost');
        $dbpasswd = $this->getConf('dbpasswd');
        // Create bloguser Object
        $bloguser = new bloguser($dbusername, $dbpasswd, $dbname, $dbhost);
        // Prüfe, ob es den angemeldeten Benutzer schon in der DB gibt, lege
ihn an wenn
        // nicht.
        $bloguser->checkBlogUser($_SERVER['REMOTE_USER']);

[...]
```

Aufgaben

(A1)

Füge die Codebestandteile von oben so in dein Projekt ein, dass diese dort funktionieren. Als angemeldeter Benutzer solltest du also die Ausgabe erhalten, dass es den Benutzer noch nicht gibt und dieser angelegt werden sollte.

(A2)

Ergänze die Klasse `bloguser` um zwei private Felder/Attribute für die ID und den Namen des Benutzers.

Implementiere in der Methode `checkBlogUser` die Funktionalität, dass der Benutzer angelegt wird, wenn er noch nicht existiert.

Hilfe Stufe 1

```
public function checkBlogUser ($dwusername) {
        print "Überprüfe, ob der Dokuwiki-Benutzer " .
$_SERVER['REMOTE_USER'] . " schon in der Datenbank angelegt ist.";

        // Datenbankabfrage wie bisher, das DBO Handle ist in $this->db
vorhanden
        // $statement= $this->db->prepare("SELECT * FROM ... WHERE ... =
:dwusername ...");
        // $statement->execute(array('dwusername' => "$dwusername"));
        // $treffer = $statement->fetch()[0];
        // (Erklärung: fetch liefert ein Array, wir holen das nullte Element
und
        /  versuchen es zu speichern. Wenn es das nullte Element nicht gibt,
ist
        // $treffer nicht gesetzt, das kann man mit 'isset' prüfen - s.u.

        // TODO: Abfrage ergänzen, um zu prüfen obs den Benutzer gibt

        if ( ! isset($treffer) ) {
            print "Nein, nicht gefunden, muss angelegt werden.";
            // TODO: Abfrage, die den User anlegt.
        }

        // Wert der Felder setzen
        $this->id = $treffer;
        $this->name = $dwusername;
```

```
    }
```

Hilfe Stufe 2

```
public function checkBlogUser ($dwusername) {
        print "Überprüfe, ob der Dokuwiki-Benutzer " .
$_SERVER['REMOTE_USER'] . " schon in der Datenbank angelegt ist.";

        $statement= $this->db->prepare("SELECT id FROM blogusers WHERE
dwusername=:dwusername");
        $statement->execute(array('dwusername' => "$dwusername"));
        $treffer = $statement->fetch()[0];

        if ( ! isset($treffer) ) {
            print "Nein, nicht gefunden, muss angelegt werden.";
            // TODO: Insert Statement vorbereiten und ausführen
            // $statement= $this->db->prepare("INSERT INTO .... ");
            // $statement->execute(array('dwusername' => "$dwusername"));

            // TODO: Nochmal die ID abfragen (die es jetzt hoffentlich gibt)
            // ....
            $treffer = $statement->fetch()[0];
        }

        // Wert der Felder setzen
        $this->id = $treffer;
        $this->name = $dwusername;
    }
```

Ergänze eine Getter-Methode, um das ID Feld des eines bloguser-Objekts zu erhalten.

(A3)

Verifiziere, dass dein Plugin sich wie gewünscht verhält, indem du in der Datenbank nachschaust, ob der Benutzer erstellt wurde.

Lege weitere Benutzer in DokuWiki an und prüfe, ob diese ebenfalls angelegt werden.

← Schritt 1Schritt 3 →

1)
und nur eine