Blogeinträge

Für den Zugriff auf die Tabelle `blogentrys`, also für das Erstellen und Anzeigen der Blogeinträge erstellen wir jetzt eine Klasse `blogentry` in der Datei `class/blogentry.class.php`[1)].

(A1)

Das Basisgerüst der `blogentry`-Klasse hat dieselbe Funktionalität wie die `bloguser`-Klasse: Im Konstruktor muss das Handle für den Datenbankzugriff "geholt" werden und als Objektvaraiable gespeichert werden.

Erstelle eine Datei mit diesem Basisgerüst, so dass in der `syntax.php` mit folgendem Code ein `blogentry`-Objekt instanziiert werden kann:

```
// Zugriff auf die Blogeinträge
$blogentry = new blogentry($dbusername, $dbpasswd, $dbname, $dbhost);
```

Teste, ob dein Code fehlerfrei funktioniert.

Hilfe

Die grundlegende `blogentry`-Klasse ist eine Kopie des Konstruktors der `bloguser`-Klasse:

```
<?php

class blogentry {
    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);

    }


}
```

```
?>
```

Methoden der Blogentry-Klasse

Jetzt müssen wie die `blogentry`-Klasse um weitere Funktionalitäten erweitern. Dazu können wir uns zunächst überlegen, welche Funktionen die Klasse benötigt, und diese als "Interface" definieren. Man kann sich vorstellen, dass ein Interface eine Klassendatei ist, die nur die Methodenköpfe ohne Code als "abstrakte Methoden" enthält:

```
<?php

interface blogentryInterface
   {
      // Alle Blogintraege anzeigen
      public function show_entries($userid=-1);

      // Neuen Blogeintrag hinzufügen
      public function insert_entry($userid);

      // Fehlaufrufe abfangen
      public function __call ($name, $param);
   }

?>
```

Die Definition der blogentry-Klasse muss nun entsprechend angepasst werden:

```
class blogentry implements blogentryInterface {
  ...
}
```

(A2)

Erstelle die Datei `class/blogentryInterface.class.php` mit den abstrakten Methoden der `blogentry`-Klasse und passe die Klassendefinition der blogentry-Klasse an. Teste das veränderte Plugin und mache dir klar, was die Fehlermeldung bedeutet und wie man sie beheben kann.

Neuer Eintrag

Um einen neuen Eintrag hinzufügen zu können, benötigen wir ein Formular, das wir z.B. mit den bereits bekannten DokuWiki-Mitteln erzeugen können:

```
public function printform() {

        // Erzeuge ein neues "Form" Objekt
        $form = new dokuwiki\Form\Form();

        $form->addHTML("<h2>Hey, was geht?!</h2>");
        $form->addTextArea('blogtext', '');
        $form->addHTML("<br/>");
        $form->addButton('submit', 'Senden');

        // Generate the HTML-Representation of the form
        return $form->toHTML();
    }
```

Wenn das Formular Daten enthält, kann muss man den Blog Eintrag hinzufügen:

```
// Neuer Eintrag!
if (isset ($_POST["blogtext"]) && checkSecurityToken() ) {
    $blogentry->insert_entry($bloguser->getID(), $_POST["blogtext"]);
    msg("Hat geklappt!");
} else {
    $renderer->doc .= $this->printform();
}
```

← Schritt 2 Schritt 4 →

1)

damit der Autoloader aus Schritt 1 sie findet…

From:
https://www.info-bw.de/ -

Permanent link:
https://www.info-bw.de/faecher:informatik:oberstufe:datenbanken:projekt:dokuwiki_plugin:microblogging:step03:start?rev=1624297995

Last update: **21.06.2021 17:53**