Projektdokumentation PHP-Blog

Severin Kaderli

Inhalt

[Einleitung & Projektbeschreibung 2](#_Toc437608500)

[Systemanforderungen 2](#_Toc437608501)

[Funktionen 2](#_Toc437608502)

[Aufbau / Struktur 3](#_Toc437608503)

[Code-Aufbau / Erklärung 3](#_Toc437608504)

[Datenbankdiagramm 4](#_Toc437608505)

# Einleitung & Projektbeschreibung

Dies ist eine Dokumentation zum Projekt im Modul 133. Die Aufgabe des Projektes war es einen Blog mithilfe von PHP zu realisieren. Die Einträge sollten entweder als Dateien oder in einer SQLite-Datenbank gespeichert werden.

# Systemanforderungen

**Wichtig:** Falls der Blog nicht funktioniert (wegen zu alter PHP-Version, nicht installierte Apache-Module), kann eine funktionierende Version auf <http://dev.severinkaderli.ch/blog> gefunden werden.

* mindestens PHP-Version 5.6
* mod\_rewrite

# Funktionen

Mein Blog verfügt über die folgenden Funktionen:

* Registrierung von neuen Benutzern
* Ein- und ausloggen von Benutzern
* Erstellen, bearbeiten und löschen von Blog-Beiträgen
* Erstellen und löschen von Kommentaren
* Benutzerverwaltung für Admins (Löschen, Benutzer zu Admins befördern)

# Verwendete Frontend-Bibliotheken

* jQuery
* Bootstrap CSS
* prism.js (Wird aktuell jedoch nicht verwendet)

# Standard-Benutzer

|  |  |
| --- | --- |
| Benutzername | Passwort |
| root | root |
| john.doe | test |

# Aufbau / Struktur

Nachfolgend ist die Ordnerstruktur des Blogs erklärt:

* Assets/
  + Dateien für das Frontend (CSS, JS, Fonts)
* Core/
  + Hauptbestandteil des Codes. Hier befinden sich die Hauptklassen die den Blog antreiben.
* Database/
  + Hier befindet sich die Datenbankdatei für SQLite.
* Views/
  + Hier befinden sich die Views die dann für den Benutzer sichtbar sind.

# Code-Aufbau / Erklärung

Um mir die Gestaltung eines PHP-Blogs einfacher zu machen, habe ich ein „kleines“ MVC-Framework erstellt. Grundsätzlich werden alle Requests per .htaccess auf die index.php weitergeleitet. Dort werden dann mit dem Router, der den Hauptteil des Frameworks ausmacht Routen definiert. Eine Route sagt eigentlich, bei welcher Anfrage, welchen Controller und welche Methode ausgeführt werden soll.

Wenn man z.B. auf die Startseite geht, wird der PostController mit der Methode index aufgerufen. Die Controller nehmen dann die Views (Templates) und übergeben ihnen die benötigten Daten (hier die Post-Objekte) damit der View korrekt angezeigt werden kann.

Im Model sind immer die Funktionen definiert um die Daten aus der Datenbank zu erhalten.

Das ist in etwa der grobe Ablauf des Frameworks. Ich habe leider ziemlich viel Zeit mit dem verbracht und konnte einige Funktionen nicht einbauen (Validator, Nachrichtenanzeige). Aber da diese nicht Bewertungsrelevant sind, habe ich diese ohne Bedenken weggelassen.

# Datenbankdiagramm

