Schritt 1: Grundfunktionalität

Zunächst soll die Grundfunktionalität implementiert werden:

- Benutzer sollen Einträge erstellen können.
- Benutzer sollen die Liste ihrer Einträge angezeigt bekommen.

Datenbankmodell:

Ein einfaches Datenbankmodell könnte zunächst so aussehen:

(A1)

Erstelle Tabellen in deiner Datenbank, die dieses Modell abbilden.

Objektorientierter Datenbankzugriff reloaded

Wenn man nun über die OOM unseres Projekts nachdenkt, macht es zunächst Sinn, dass man zwei Klassen verwenden möchte:

- `bloguser.class.php`: Alle Operationen, die die Benutzer betreffen. Vor allem „erbt“ unser Blog die Benutzer des DokuWiki Systems, d.h. wenn ein Benutzer zum erstem Mal einen Eintrag verfassen möchte, muss dieser wenn nötig zunächst der Benutzertabelle des Blogs hinzugefügt werden.
- `blogentry.class.php`: Alle Operationen, die die Blogeinträge betreffen.

Beide Klassen müssen auf die Datenbank zugreifen.

(A2)

- Könnte man für die Grundfunktionalität auch mit einer Datenbanktabelle und einer Klasse auskommen?
- Warum könnte das das hinsichtlich der Erweiterbarkeit des Projekts ungeschickt sein, so zu

beginnen?

Refactoring des bisherigen Beispielplugins

Um deinen Plugin-Code besser zu strukturieren und auf die kommenden Anpassungen vorzubereiten, gehe wie folgt vor:

- Erstelle einen Commit mit einen aktuellen Änderungen, so dass dein Arbeitsverzeichnis sauber ist. Erzeuge dann einen neuen *Branch* mit dem Befehl `git checkout -b „microblog“`. Schau dir das Ergebnis mit dem Befehl `git branch` an, es sollte in etwa so aussehen:

<html> <script id=„asciicast-qUmzyNi9H9FKQQNdux5xraFEE“ src=„https://asciinema.org/a/qUmzyNi9H9FKQQNdux5xraFEE.js“ async></script> </html>

- Lege ein Unterverzeichnis `class` im Plugin-Ordner an.
- Verschiebe die Datei mit deiner Datenbank-Klasse in diesen Ordner und benenne sie um in `mysqldb.class.php`
- Passe das `include` Statement in der `syntax.php` entsprechend an.
- Teste, ob der bisherige Stand der Entwicklung noch immer funktioniert.

Automatisches Laden benötigter Klassendefinitionen

Wenn künftig weitere Klassendefinitionen zu unserem Projekt hinzukommen, wird es immer aufwändiger, alle benötigten Dateien von Hand im Kopf der Steuerklasse[1)] einzubinden.

Man kann das durch einen „Autoloader“ erledigen lassen. Damit das klappt, müssen ein paar Konventionen eingehalten werden:

- Alle Klassen (und Interfaces) müssen in einer Datei `class/<NAME>.class.php` definiert werden.
- Die in den Dateien definierten Klassen müssen genau so heißen, wie die Dateien. Man darf also nicht die Klasse `mysqldb` in einer Datei mit dem Namen `db.class.php` definieren, sondern diese muss in der Datei `mysqldb.class.php` definiert werden.

1)

unser syntax.php