Schritt 1: Grundfunktionalität

Zunächst soll die Grundfunktionalität implementiert werden:

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

Ein erstes Datenbankmodell

Ein einfaches Datenbankmodell könnte zunächst so aussehen:

(A1)

Erstelle Tabellen in deiner Datenbank, die dieses Modell abbilden.

Mehr Klassen bitte

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 Ordner deines Plugins 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.

Wenn man sich an diese Regeln hält kann man anstelle des `include` Statements zu Beginn der `syntax.php` folgenden Code einfügen:

```
// Klassendateien mit Hilfe eine anonymen Funktion als
// Autoloader einbinden
spl_autoload_register(function ($class)
   {
    // DokuWiki arbeitet stets in seinem DocRoot, wir suchen die Dateien
    // aber in einem UVZ des Plugin-Verzeichnisses -anpassen wenn das
    // Plugin nicht "projekt" heißt!
    $plugin_dir=DOKU_PLUGIN . "/projekt/";

    // Wenn eine Klasse oder ein Interface noch nicht existiert
    // soll versucht werden, die passende Datei zu laden.
```

```
    if(!class_exists($class) or !interface_exists($class))
        {
            $file = $plugin_dir . 'class/' . $class . '.class.php';
            // Wenn die Datei nicht existiert, Fehler ausgeben.
            if(!file_exists($file))
               die ('Fehler beim automatischen Laden einer Klassendatei.<br
/>Es wurde erfolglos versucht die Klasse ' . $class . ' aus der Datei ' .
$file . ' zu laden.');
            // Wenn die Datei existiert, laden
            require_once($file);
        }
   });
```

(A3)

Ersetze das Include-Statement durch den Code oben und versuche zu verstehen, was dieser macht.

Teste ob der Code tatsächlich automatisch Klassendateien nachlädt, indem du einen Fehlerfall, erzeugst: versuche dazu ein Objekt zu erzeugen, für das es keine Klassendefinition gibt, beispielsweise `$benutzer = new bloguser();`.

Übersicht Schritt 2 →

---

1)

unsere syntax.php