Webentwicklung (WE) - FWPM

# Übung 6: “Nutzerverwaltung und Security”

In dieser Übung geht es darum, das eigene Projekt um die Möglichkeit des Logins zu erweitern Das erlaubt z.B. Inhalte nur von authentifizierten Nutzern erstellbar zu machen. Auf eine Rechteverwaltung wird in dieser Übung bewusst verzichtet um den Umfang kleiner zu halten. In der Praxis ist eine differenzierte Rechteverwaltung allerdings sehr sinnvoll.

## 1. Login ermöglichen

In der ersten Aufgabe soll ein simpler Login ermöglicht werden. Er besteht aus den Komponenten: Login-Formular, Controller und der Möglichkeit zum Logout.

Erstelle einen Controller der sich um den Login kümmert.

a. Erstelle einen AuthController der neben dem Login mehrere Actions wie z.B. auch den Logout bereitstellen kann.
b. Der Controller muss unterscheiden können ob er eine View zum Rendern des Login Formulars nutzt oder einen Loginversuch verarbeiten soll. Im einfachsten Fall überprüfst du ob der Request bereits gesendete Formulardaten enthält. Ist das nicht der Fall müssen wir das Login Formular anzeigen.

Erstelle eine View Klasse für das Login Formular und ein entsprechendes Formular innerhalb eines HTML Templates.

c. Die View Klasse hat die Aufgabe das Template zu rendern und vor allem beim Login sinnvolles Feedback zu Fehleingaben zu geben. Feedback an den Nutzer ist eine komplexe Angelegenheit, aber in dieser Übung reicht an dieser Stelle eine einfache Fehlermeldung z.B. über eine Exception.
d. Das Template ist ein einfaches Formular mit den zwei Eingabefeldern “Username/E-Mail” und “Passwort” und dem Absendebutton. Als Methode sollte POST verwendet werden.

Erhält der in 1. erstellte Controller jetzt die Formulardaten muss er den Login prüfen können. Halte dich dazu an die Codebeispiele aus der Vorlesung.

e. Validierung des Inputs auch im Controller nicht vergessen! Login Daten werden eigentlich immer mit der Datenbank abgeglichen und sind ein einfaches Ziel für SQL Injections.
f. Um die noch nicht implementierten Teile der Persistenzschicht zu umgehen kannst du die Abfrage im Repository für einen schnellen Test auskommentieren und die *password_verify()* Methode durch ein *true* ersetzen. Das erleichtert dir dich zuerst auf den Login zu konzentrieren.
g. Überprüfen deinen Erfolg. Nutze dazu die Entwicklertools deines Browsers um die Cookies zu überprüfen. Im Mozilla Firefox sind Cookies unter “Web-Speicher” zu finden, im Google Chrome unter “Application->Storage”.

**Warnung:** Wenn du für einen schnellen Test Logik "überbrückst" dann diesen Stand auf keinen Fall per commit versionieren! Jeder commit sollte ausschließlich sicheren Code enthalten.

# 3. Datenmodell für Nutzer

1. Um überhaupt einen Login nutzen zu können, braucht es einen Nutzer. Überlege dir welche Felder ein Nutzer Model braucht und erstelle die Klasse.
    a. Username und Passwort reichen in der Regel aus. Ist der Username gleichzeitig die E-Mail Adresse lässt sie sich bei der Eingabe validieren und man hat einen Kommunikationskanal zu seinen Nutzern.
    b. Erstelle ein Datenbankschema passend zu deiner existierenden Nutzer Model Klasse.
    a. Achte auf eine passende Länge für das Passwort Feld, da zukünftige Algorithmen längere Hashes generieren könnten. 255 Zeichen sollten es schon sein.
    b. Erstelle einen User in deiner neue Datenbank Tabelle. Das Passwort kannst du auf der Kommandozeile hashen: *php -r "var_export(password_hash('<DEIN_PASSWORT_HIER>', PASSWORD_DEFAULT));"* Es ist gängige Praxis, dass Anwendungen mit einem Account in Datenbank ausgeliefert werden bzw. dieser beim Setup erzeugt wird.
2. Erstelle die Entität bzw Model Klasse deines Nutzers.
3. Implementiere eine Repository Klasse um deinen Nutzer aus der Datenbank abzurufen, und ihn bei Bedarf speichern zu können.
    a. Um die Authentifikation z.B. einer Nutzerin anhand ihres Usernamens zu ermöglichen muss das Repository eine Methode wie z.B. *getByUsername($username)* bieten.
    b. Integriere deine Repository Klasse in deinen AuthController

# 2. Login nutzen

Der Login funktioniert. Jetzt geht es darum ihn sinnvoll in die Anwendung zu integrieren. Dazu gehört sowohl die Implementierung einer Zugangskontroller für als geschützt geplante Routen und Funktionen, als auch einen bewussten Logout zu ermöglichen.

1. Überlege dir wozu du den Login eingebaut hast. Welche Funktionen deiner Anwendung sollen nur angemeldeten Nutzern zur Verfügung stehen? In der weiteren Aufgabe gehen wir von einer Route zum Erstellen von Blogposts aus.
2. Implementiere die Überprüfung der erfolgreichen Authentifikation innerhalb des Controllers.
    a. Wähle eine Stelle die die Ausführung im Controller ohne Überprüfung auf jeden Fall verhindert. Z.B. ganz zu Beginn deiner Action Methoden oder innerhalb einer separaten Methode die du im Konstruktor aufrufen kannst.
    b. Orientiere dich an dem Implementierungsvorschlag aus der Vorlesung wenn du nicht weiterkommst.
    c. Stelle auch sicher, dass du eine Ausgabe für nicht korrekt authentifizierte Nutzer auf jeden Fall ausschließt. Ein Redirect allein reicht nicht.
3. Implementiere eine zentrale Logout Funktion.
    a. Erstelle dazu am Besten eine neue Route und eine Action in deinem AuthController.

b. Zum Logout musst du nur die Session des Nutzers zerstören. Nutze dazu die native *session_destroy()* Funktion.

**Bonus:** Lagere die Authentifikation so aus, dass du sie an unterschiedlicher Stelle wieder verwenden kannst. Für Controller macht z.B. ein abstrakter Controller Sinn, von dem Controller geschützter Routen dann nur noch erben müssen oder eine zentrale Kontrolle innerhalb des Routers.