Webentwicklung (WE) - FWPM

Übung 2: “OOP und Grundstruktur”

Objektorientierte Programmierung

Deine Anwendung soll so wenig wie möglich auf Skriptlogik zurückgreifen. Daher werden wir jede Logik innerhalb von Klassen abbilden. Dazu muss dein Composer Autoloading richtig konfiguriert sein und deine Klasse nach dem PSR-4 Standard benannt und mit Namespaces versehen sein.

1. Informiere dich über die Nutzung von Composer und seinem Autoload Feature.
    a. Konfiguriere deine *composer.json* korrekt und erstelle eine kleine “Hello World” Testklasse um zu sehen ob sie instanziiert werden kann

Request/Response

Um eine klare Kontrolle über den Datenfluss innerhalb unserer Anwendung zu erhalten, werden wir die globalen magischen Konstanten $_SERVER und $_REQUEST in DTOs einbinden.
So sind wir in der weiteren Verarbeitung innerhalb unserer Anwendung nicht mehr auf die globalen magischen Konstanten angewiesen.
Selbes gilt für die eine Response Klasse, über die die Ausgabe kontrolliert werden kann.

1. Erstelle eine Klasse *Request* die Zugriff auf die globalen magischen Konstanten $_SERVER und $_REQUEST hat.
    a. Deine Requestklasse sollte separiert Daten über die angefragte URL, die genutzte HTTP Request Method, die vorhandenen Request Header und eventuelle GET oder POST Parameter bereitstellen können
    b. Schaffe Methoden, über die andere Klassen diese Daten aus einer Instanz deiner Requestklasse abfragen können.
        i. Es bietet sich an zu separieren zwischen einer Abfrage ob ein Wert vorhanden ist (*hasXyz()*) und der Abfrage des Werts selbst (*getXyz()*)
    c. Stelle sicher, dass $_SERVER und $_REQUEST sonst nirgends mehr in deiner Anwendung verwendet werden.
        i. Alle Information die deine Anwendung über den Request hat muss ab jetzt aus einer Instanz deiner Request Klasse kommen
        ii. Instanziere deine Request Klasse einmal in der *index.php* und reiche diese Instanz bei Bedarf weiter. Instanziere nicht neu

2. Baue eine Klasse namens *Response* die mindestens die Ausgabe als String (man spricht vom Response Body) als auch die Möglichkeit zum Setzen des HTTP Statuscodes hat.
    a. Anders als die Request Klasse, braucht die Response Klasse neben Methoden zur Abfrage dieser Daten auch Methoden um sie zu setzen
    b. Instanziere deine Response Klasse einmal innerhalb deiner index.php und reiche diese Instanz zur Verarbeitung weiter sobald Controller implementiert wurden.
        i. Den Response Body deiner Instanz kannst du zentral am Ende deiner index.php ausgegeben (ein *echo* reicht schon).
        ii. Den Status Code der vom Webserver kommt, kannst du mit der Funktion *http_response_code()* global setzen.

Routing

Du hast bereits aus Übung 1 zwei separate “Features” in deiner Anwendung, sie kann auf verschiedene URLs unterschiedlich reagieren. Die Möglichkeit je nach URL anders zu reagieren nennt man Routing. Der Request wird innerhalb deiner Anwendung sinnvoll weitergereicht.
In diesem Schritt wird dieses Routing in eine extra Klasse integriert. Einen Router

1. Erstelle eine Klasse die das Routing innerhalb deiner Anwendung übernimmt.
    a. Parse die nötigen Daten aus deinem Request Objekt. Es hat die Informationen über die aufgerufene URL
    b. Überlege dir ein Muster nach dem du an deine späteren Controller weiterleitest. Willst du URLs nach einem gewissen Muster weiterleiten (siehe Vorlesung)? Oder sollen die Routen fest konfiguriert werden, z.B. über eine *addRoute(string $routePath, $action)* Methode im Router
2. Überprüfe ob beide “Features” deiner Anwendung unter den für sie vorgesehenen Routen erreichbar sind.

Bonus: Kapsele die Ausgabe für verschiedene Routen in extra Klassen. Sie werden später deine jeweiligen Controller werden.