BKBdemy /htdocs Dokumentation

.htaccess-Datei Dokumentation

Die .htaccess-Datei ist eine Konfigurationsdatei für Webserver, die es ermöglicht, verschiedene Einstellungen für die Website auf Verzeichnis- oder Dateiebene zu definieren. In der folgenden Dokumentation wird die .htaccess-Datei für die BKBdemy-Website beschrieben.

BEGIN bkbdemy

<IfModule mod_rewrite.c> RewriteEngine On

Exclude certain file types from rewrite rules

RewriteRule
.(css|js|png|gif|jpg|jpeg|woff|ttf|svg|ico|pdf|doc|docx|xls|xlsx|ppt|pptx|zip
|rar|mp4|avi|mov|wmv|flv|mp3|wav|ogg|webm|json)$ - [L]

Redirect HTTP requests to HTTPS

RewriteCond %{HTTPS} !=on RewriteRule ^
https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

Handle all requests with index.php

RewriteCond %{REQUEST_FILENAME} !-f RewriteCond %{REQUEST_FILENAME}
!-d RewriteRule ^(.*)$ index.php [L] </IfModule>

END bkbdemy

Diese .htaccess-Datei enthält einige Einstellungen und Regeln, die für die BKBdemy-Website relevant sind:

- RewriteEngine On: Diese Einstellung aktiviert die Apache Rewrite Engine, die es ermöglicht, URLs innerhalb der Website umzuschreiben und weiterzuleiten.
- RewriteRule
\.(css|js|png|gif|jpg|jpeg|woff|ttf|svg|ico|pdf|doc|docx|xls|xlsx|ppt|p
ptx|zip|rar|mp4|avi|mov|wmv|flv|mp3|wav|ogg|webm|json)$ - [L]:
Diese Regel schließt bestimmte Dateitypen von den Rewrite-Regeln aus.

Das bedeutet, dass URLs mit diesen Dateitypen nicht umgeschrieben werden.

- RewriteCond %{HTTPS} !=on: Diese Bedingung prüft, ob die Anfrage nicht bereits über HTTPS erfolgt, und leitet sie gegebenenfalls an HTTPS weiter.
- RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]: Diese Rewrite-Regel leitet die Anfrage an die HTTPS-Version der Website weiter und gibt einen 301-Statuscode zurück, um den Suchmaschinen mitzuteilen, dass sich die URL dauerhaft geändert hat.
- RewriteCond %{REQUEST_FILENAME} !-f: Diese Bedingung prüft, ob die angeforderte Datei nicht existiert.
- RewriteCond %{REQUEST_FILENAME} !-d: Diese Bedingung prüft, ob das angeforderte Verzeichnis nicht existiert.
- RewriteRule ^(.*)$ index.php [L]: Diese Rewrite-Regel leitet alle Anfragen an die index.php-Datei weiter, die für die Verarbeitung der Website-URLs und -Anfragen zuständig ist.

Es ist wichtig, dass die .htaccess-Datei korrekt konfiguriert ist, um sicherzustellen, dass die Website korrekt funktioniert. Änderungen an der .htaccess-Datei sollten immer vorsichtig und mit Bedacht vorgenommen werden, um sicherzustellen, dass keine unerwarteten Auswirkungen auftreten.

Routing Dokumentation

Die Route-Klasse stellt ein einfaches Routing-System dar, das es ermöglicht, Anfragen an bestimmte URLs an entsprechende Funktionen weiterzuleiten.

- Die statische Variable $routes ist ein Array, das alle definierten Routen enthält. Jede Route besteht aus einem regulären Ausdruck ($expression), der die URL-Muster beschreibt, einer Callback-Funktion ($function), die ausgeführt wird, wenn die URL-Muster übereinstimmt, und einer HTTP-Methoden-Definition ($method), die die HTTP-Methoden festlegt, auf die die Route reagieren soll.
- Die Methoden add, pathNotFound und methodNotAllowed fügen neue Routen hinzu und definieren Funktionen, die aufgerufen werden sollen, wenn die angeforderte URL nicht gefunden wird oder die angeforderte HTTP-Methode nicht unterstützt wird.
- Die run-Methode ist die Hauptmethode der Klasse, die aufgerufen wird, wenn eine neue Anfrage an den Server gesendet wird. Sie iteriert über alle definierten Routen, sucht nach einer Übereinstimmung des URL-Musters und der HTTP-Methoden-Definition und führt die zugehörige

Callback-Funktion aus, wenn eine Übereinstimmung gefunden wurde. Wenn keine Übereinstimmung gefunden wurde, wird entweder eine 404-Fehlermeldung ausgegeben oder die Funktion aufgerufen, die für nicht gefundene Pfade definiert wurde, oder eine 405-Fehlermeldung ausgegeben oder die Funktion aufgerufen, die für nicht unterstützte HTTP-Methoden definiert wurde.

Das Routing-System bietet eine einfache Möglichkeit, den Code einer Webseite zu organisieren und die Verarbeitung von Anfragen auf verschiedene Funktionen zu verteilen.

Webpack Dokumentation

Dies ist eine Konfigurationsdatei für das Build-Tool Webpack, die es Entwicklern ermöglicht, eine JavaScript-App zu erstellen und zu optimieren.

Zunächst wird der Einstiegspunkt (entry) für die App festgelegt. In diesem Fall ist es ein JavaScript-Modul (index.js) im Ordner /src/js. Der Ausgabepunkt (output) für die optimierte App ist ein JavaScript-Datei (main.min.js) im Ordner /dist.

In der Modulregel (module.rules) wird definiert, wie verschiedene Arten von Dateien verarbeitet werden sollen. In diesem Fall wird für JavaScript-Dateien der Babel-Loader verwendet, um es mit älteren Browsern kompatibel zu machen. Für CSS-Dateien wird das MiniCssExtractPlugin-Plugin verwendet, um eine separate CSS-Datei (main.min.css) zu erstellen. Das SASS-Loader wird verwendet, um SASS-Dateien in CSS zu kompilieren. Und für die jQuery-Bibliothek wird der Expose-Loader verwendet, um sie global verfügbar zu machen.

Verschiedene Plugins werden ebenfalls verwendet, um den Build-Prozess zu optimieren und zu verwalten. Dazu gehören das ProvidePlugin-Plugin, das das globale Verwenden von jQuery in der App ermöglicht, das CleanWebpackPlugin-Plugin, das den Ausgabepunkt vor jedem Build löscht, das MiniCssExtractPlugin-Plugin, das CSS-Dateien in eine separate Datei extrahiert, und das CleanTerminalPlugin-Plugin, das das Terminal vor jedem Build löscht.

Die Optimierungseinstellungen (optimization) sind für das Minimieren des Codes verantwortlich. Dazu werden der TerserPlugin-Plugin und der CssMinimizerPlugin-Plugin verwendet.

Schließlich werden einige Einstellungen für das Entwickeln (devtool, watch, watchOptions) festgelegt, um einen schnellen Entwicklungsprozess zu ermöglichen. Die Mode-Eigenschaft ist auf "production" gesetzt, um sicherzustellen, dass die App optimal kompiliert wird.

Login / Logout / Registrierung Dokumentation #

Die Funktionen initLogin(), initLogout() und initRegistration() kümmern sich um den Login, Logout und die Registrierung von Benutzern auf der Website.

initLogin():

Diese Funktion ist für den Login von Benutzern zuständig. Es werden verschiedene Variablen für die Eingabefelder, die Schaltfläche zum Einreichen des Formulars und das Behältnis für Fehlermeldungen definiert. Wenn der Benutzer auf die Schaltfläche klickt, werden die Eingabedaten in ein JSON-Objekt gepackt und an die API-Route /auth/login gesendet. Wenn die API antwortet, wird geprüft, ob ein gültiger Token zurückgegeben wurde. Wenn dies der Fall ist, wird der Token im Local Storage gespeichert und der Benutzer auf die Dashboard-Seite weitergeleitet. Wenn kein Token zurückgegeben wurde, wird dem Benutzer eine Fehlermeldung angezeigt.

initLogout():

Diese Funktion ist für den Logout von Benutzern zuständig. Wenn der Benutzer auf die Schaltfläche "Logout" klickt, wird eine Anfrage an die API-Route /auth/logout gesendet, um den aktuellen Token ungültig zu machen. Wenn die API antwortet, wird der Token aus dem Local Storage entfernt und der Benutzer wird auf die Startseite weitergeleitet.

initRegistration():

Diese Funktion ist für die Registrierung von Benutzern zuständig. Es werden verschiedene Variablen für die Eingabefelder, die Schaltfläche zum Einreichen des Formulars und das Behältnis für Fehlermeldungen definiert. Wenn der Benutzer auf die Schaltfläche klickt, werden die Eingabedaten in ein JSON-Objekt gepackt und an die API-Route /auth/register gesendet. Wenn die API antwortet, wird geprüft, ob ein gültiger Token zurückgegeben wurde. Wenn dies der Fall ist, wird der Token im Local Storage gespeichert und der Benutzer auf die Dashboard-Seite weitergeleitet. Wenn kein Token zurückgegeben wurde, wird dem Benutzer eine Fehlermeldung angezeigt.

Es gibt auch Hilfsfunktionen, die in diesen Funktionen verwendet werden, wie animateErrorMessage(), getToken() und getUserData(). animateErrorMessage() fügt eine CSS-Klasse zu einem Fehlermeldungselement hinzu und entfernt sie nach einer bestimmten Zeit wieder. getToken() ruft den aktuellen Token aus dem Local Storage ab und gibt ihn zurück, oder gibt null zurück, wenn kein Token gefunden wurde. getUserData() sendet eine Anfrage an die API-Route /auth/me und gibt die Daten des aktuellen Benutzers zurück, wenn ein gültiger Token gefunden wurde.