WEB-TECHNOLOGIEN

Praktikum 12:
Node.js und Express.js
(und CTF)

ABNAHME 3 (BONUSPUNKTE)

Termin: Kalenderwoche 4, also 21.01., 22.01., 24.01. und 25.01.19

- Abgenommen werden die Ergebnisse von Praktikum 9, 10, 11, 12
- Bis zu 4 Bonuspunkte

Organisatorisches:

- Matrikelnummer/FHCard bereithalten
- Bonuspunkte werden in SmartAssign eingetragen (Login mit FH-Kennung): https://smartassign.inf.fh-dortmund.de/

Aus technischen Gründen gilt in SmartAssign: 1 Punkt = 0,1 Bonuspunkte! Beispiel: 45 Punkte in SmartAssign bedeuten 45/10=4,5 Bonuspunkte.

RAHMENAUFGABE: RAUMVERWALTUNG

- Im Rahmen des Praktikums entwickeln wir eine Web-Anwendung zur Verwaltung von Raumbuchungen (z.B. in einer Fachhochschule ;-)).
- Diese Web-Anwendung werden wir Schritt für Schritt mit weiteren Anforderungen, Funktionen und Technologien erweitern.

Verwenden Sie in den Laboren stets das Laufwerk "H:" zur Speicherung Ihrer Daten, damit Sie auf diese auch später (z.B. an einem anderen Laborrechner) noch zugreifen können.

Legen Sie Ihren Quellcode pro Praktikumsaufgabe in separaten Verzeichnissen (z.B. „praktikum2“, „praktikum3“) ab, damit die einzelnen Entwicklungsschritte bei den Abnahmen ersichtlich sind.

ALLGEMEINE HINWEISE ZU PRAKTIKUM 12

- Bei dem hier vorliegenden Praktikum 12 handelt es sich um die letzte Praktikumsaufgabe. Diese hat einen größeren Umfang als die bisherigen Aufgaben.

Für die Bearbeitung des Praktikums haben Sie dementsprechend ca. zwei Wochen Zeit!

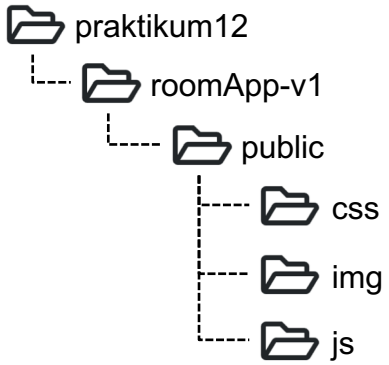
- In Praktikum 12 werden wir unsere Raumverwaltungsanwendung mit Node.js in zwei Varianten implementieren:
  - *RoomApp-V1*: Realisierung mit Hilfe der Module „fs“ und „http“
  - *RoomApp-V2*: Realisierung mit Hilfe von Express.js und EJS

Praktikum 12: ROOMAPP-V1

ERSTE VERSION (V1) DER ANWENDUNG

1. Strukturierung des Projekts:

a. Erstellen Sie für das Praktikum ein neues Verzeichnis „praktikum12“ und legen Sie darin die folgende Verzeichnisstruktur an:



- „roomApp-v1“ ist dabei das Wurzelverzeichnis für diese Version der Anwendung.
- Im Verzeichnis „public“ befinden sich statische Ressourcen, die an Clients ausgeliefert werden sollen.

b. Kopieren Sie nun die Dateien aus Praktikum 11 wie folgt in die neue Verzeichnisstruktur:

1. Alle `.css`-Dateien in das Verzeichnis „roomApp-v1/public/css“,
2. alle Bilddateien in das Verzeichnis „roomApp-v1/public/img“,
3. alle `.js`-Dateien in das Verzeichnis „roomApp-v1/public/js“, und
4. alle `.html`-Dateien (***außer*** der Seite für die *Liste der verfügbaren Räume*) in das Verzeichnis „roomApp-v1/public“.

Praktikum 12: ROOMAPP-V1

2. Web-Server:

Implementieren Sie nun einen Web-Server, der bei Aufruf durch den Browser unsere Web-Anwendung ausliefert. Gehen Sie dabei wie folgt vor:

a. Realisieren Sie den Web-Server in einer neuen Datei „roomApp-v1/server.js“.

b. Verwenden Sie die Node.js-Module „fs“ und „http“.

c. Sorgen Sie dafür, dass der Web-Server unter der URL http://localhost:8020 erreichbar ist.

d. Beim Eintreffen einer HTTP-GET-Anfrage soll sich der Web-Server nun folgendermaßen verhalten:

- Wird in der URL der HTTP-GET-Anfrage eine (beliebige!) Datei angefordert, so soll der Web-Server versuchen, diese Datei aus dem Verzeichnis „public“ zu lesen und mit einem entsprechenden HTTP-Statuscode in der HTTP-Antwort zurückzuliefern. *Beispiel:* Wird im Browser die URL http://localhost:8020/raumdetail.html eingegeben, so soll der Web-Server versuchen, die Datei „public/raumdetail.html“ einzulesen und in der HTTP-Antwort an den Browser zu senden.
- Wird eine Datei angefordert, die im Verzeichnis „public“ (oder darin enthaltenen Unterverzeichnissen) nicht existiert, so soll der Web-Server den Statuscode 404 (NOT FOUND) sowie eine vordefinierte Fehlerseite zurückliefern. Erstellen Sie zu diesem Zweck eine HTML-Seite „404.html“ und legen Sie diese im Verzeichnis „public“ ab.

Praktikum 12: ROOMAPP-V1

e. Sonderfall: Sorgen Sie dafür, dass bei Eingabe der URL http://localhost:8020 (d.h. ohne Anforderung einer konkreten Datei) immer standardmäßig die HTML-Seite für das Dashboard zurückgeliefert wird.

f. Passen Sie in den HTML-Dateien sämtliche Referenzen auf Bilder, CSS-Dateien, JavaScript-Dateien sowie Verlinkungen auf andere HTML-Dateien an, so dass diese ebenfalls fehlerfrei ausgeliefert werden.

Hinweise:

- Beachten Sie die Vorlesungsfolien *52.1* und *52.2* für den Umgang mit dem `Request`-Objekt.
- Mit Hilfe der (in Node.js standardmäßig verfügbaren) Variable `__dirname` (zwei Unterstriche) erhalten Sie Zugriff auf das Verzeichnis, in welchem sich die Datei „server.js“ befindet (über die wir unseren Web-Server ausführen).
- Zielzustand: Wenn Sie alles richtig implementiert haben, dann sollte sich die Web-Anwendung so verhalten und auch so aussehen wie bisher, d.h.:
  - die Bilder und CSS-Layouts sollten angezeigt werden
  - die Navigation zwischen den HTML-Seiten sollte funktionieren, und
  - die clientseitigen JavaScript-Dateien sollten weiterhin funktionieren (z.B. Plus-Kachel auf dem Dashboard, Erzeugung der *Detailseite zu einem Raum* per DOM-Manipulation etc.)

Praktikum 12: ROOMAPP-V1

3. Liste der verfügbaren Räume dynamisch erzeugen:

Unser Web-Server soll nun exemplarisch die HTML-Seite mit der *Liste der verfügbaren Räume* dynamisch erzeugen. Alle weiteren Dateien sollen wie bisher statisch aus dem „public“-Verzeichnis ausgeliefert werden. Gehen Sie dabei wie folgt vor:

a. Kopieren Sie den in Praktikum 10 erstellten JavaScript-Code zur Definition eines „Raum“-Objektes in die Datei „server.js“.

b. Erstellen Sie ein Array mit einigen beispielhaften „Raum“-Objekten – dieses soll uns als „Fake-Datenbank“ dienen.

c. Erweitern Sie den Web-Server nun folgendermaßen:

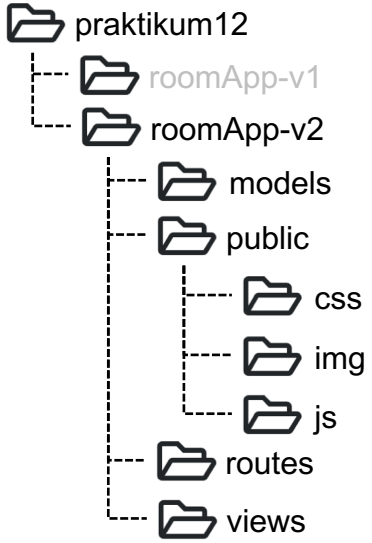
- Bei Aufruf der URL http://localhost:8020/raumliste.html soll der Web-Server nicht versuchen, die Datei “public/raumliste.html“ auszuliefern. Stattdessen soll der entsprechende HTML-Code dynamisch erzeugt werden.
- Verwenden Sie zu diesem Zweck, wie in der Vorlesung gezeigt, die Template-Literale von JavaScript. Erzeugen Sie die resultierende Liste auf Basis des oben erstellten Arrays von Raum-Objekten.
- Geben Sie den HTML-Code nach erfolgter Erzeugung in der HTTP-Antwort zurück.
- Die zurückgelieferte HTML-Seite mit der *Liste der verfügbaren Räume* soll im Browser aussehen wie bisher (Layout, Bilder etc.).

Praktikum 12: ROOMAPP-V2

ZWEITE VERSION (V2) DER ANWENDUNG

1. Strukturierung des Projekts:

a. Erstellen Sie für die zweite Version der Anwendung ein neues Verzeichnis „roomApp-v2“ und legen Sie darin die folgende Verzeichnisstruktur an:



b. Kopieren Sie erneut folgende Dateien aus Praktikum 11 wie folgt in die neue Verzeichnisstruktur (Achtung, diesmal **nicht** die HTML-Dateien!):

1. Alle `.css`-Dateien in das Verzeichnis „roomApp-v2/public/css“,
2. alle Bilddateien in das Verzeichnis „roomApp-v2/public/img“, und
3. alle `.js`-Dateien in das Verzeichnis „roomApp-v2/public/js“.

Praktikum 12: ROOMAPP-V2

2. Initialisierung des Projekts:

a. Erzeugen Sie mit Hilfe von NPM eine „package.json“-Datei im Verzeichnis „roomApp-v2“. Legen Sie dabei als „Entry Point“ der Anwendung die Datei „app.js“ fest.

b. Installieren Sie mit Hilfe von NPM im Verzeichnis „roomApp-v2“ Express und EJS:

```
$ npm install --save express
$ npm install --save ejs
```

Diese sollten anschließend in der „package.json“-Datei im Abschnitt `dependencies` zu finden sein. Zudem sollte Ihr Projekt nun ein neues Verzeichnis „node_modules“ enthalten.

Praktikum 12: ROOMAPP-V2

3. Modulare Umsetzung der Anwendung:

Realisieren Sie nun die Web-Anwendung mit Hilfe von Express und EJS. Gehen Sie wie folgt vor:

a. Sorgen Sie dafür, dass die Anwendung unter der URL http://localhost:8040 erreichbar ist.

b. Realisieren Sie die Datei “roomApp-v2/app.js“ als Einstiegspunkt für die Anwendung. Hier können Sie auch benötigte Einstellungen vornehmen.

c. Die Anwendung soll folgende URLs/Routen unterstützen:
   a. http://localhost:8040 → Dashboard
   b. http://localhost:8040/raumliste → Liste der verfügbaren Räume
   c. http://localhost:8040/raumdetail → Detailseite zu einem Raum
   d. http://localhost:8040/buchungsdetail → Detailseite zur Buchung
   e. http://localhost:8040/neuebuchung → Formular zum Anlegen einer Buchung
   f. Statische Inhalte (Bilder, CSS, clientseitiges JavaScript) sollen aus dem Verzeichnis „public“ (bzw. aus darin enthaltenen Unterverzeichnissen) heraus ausgeliefert werden.
   g. Für nicht existierende URLs und Ressourcen soll wie auch bei RoomApp-V1 wieder der Statuscode 404 und eine Fehlerseite zurückgeliefert werden.

Praktikum 12: ROOMAPP-V2

d. Lagern Sie die Logik für das Routing in ein eigenes Modul „roomApp-v2/routes/routes.js“ aus.

e. Realisieren Sie alle HTML-Seiten Ihrer Web-Anwendung (auch die Fehlerseite!) als EJS-Templates und legen Sie diese im Verzeichnis „views“ ab. Tipp: Verwenden Sie die HTML-Dateien der RoomApp-V1 als Vorlage.

f. Lagern Sie die Bereiche, die sich auf allen HTML-Seiten wiederholen (Kopfbereich mit Navigation, Fußbereich und Bereich für aktuelle Ankündigungen), in eigene EJS-Templates aus. Verwenden Sie `include`, um diese Bereiche in allen Seiten jeweils an den richtigen Stellen einzubinden.

Praktikum 12: ROOMAPP-V2

4. Liste der verfügbaren Räume dynamisch erzeugen:

Wie auch schon bei RoomApp-V1 soll die Seite mit der *Liste der verfügbaren Räume* dynamisch erzeugt werden. Gehen Sie wie folgt vor:

a. Kopieren Sie den JavaScript-Code zur Definition eines „Raum“-Objektes sowie das Array mit den beispielhaften „Raum“-Objekten aus RoomApp-V1 und lagern Sie diesen in ein eigenes Modul „roomApp-v2/models/rooms.js“ aus.

b. Sorgen Sie dafür, dass das Array der “Raum“-Objekte dem EJS-Template für die *Liste der verfügbaren Räume* beim Rendern zur Verfügung steht. Nutzen Sie EJS, um die Liste auf Basis der “Raum“-Objekte zu erzeugen.

Hinweis:

Zielzustand: Wenn Sie alles richtig implementiert haben, dann sollte sich die Web-Anwendung so verhalten und auch so aussehen wie bisher, d.h.:

- die Bilder und CSS-Layouts sollten angezeigt werden
- die Navigation zwischen den HTML-Seiten sollte funktionieren, und
- die clientseitigen JavaScript-Dateien sollten weiterhin funktionieren (z.B. Plus-Kachel auf dem Dashboard, Erzeugung der *Detailseite zu einem Raum* per DOM-Manipulation etc.)

Zusatzaufgabe 3: Capture the Flag (CTF)

Die folgende Aufgabe ist freiwillig – es gibt für diese Aufgabe keine Bonuspunkte (nur Ruhm, Ehre, Spaß, etc. pp.)!

In dieser freiwilligen Zusatzaufgabe gibt es eine neue Challenge vom Typ „Capture the Flag“ (CTF, siehe Praktikum 5 für die grundlegende Beschreibung des Prinzips). Auch dieses Mal ist das Flag wieder eine Zeichenkette, die mit „ctf_“ beginnt (z.B. `ctf_This_i5_the_Flag_forma7`).

Finden Sie das Flag, das hier versteckt ist:

Challenge 3: „Zwitscher-Hashflag“ (Schwierigkeitsgrad: mittel)

#ctfftwwtfscnr

Wenn Sie das Flag gefunden haben, schicken Sie es per E-Mail an sven.joerges@fh-dortmund.de. Und nichts verraten! ☺