EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Informationsdienste – Prof. Dr. Thomas Walter

Grundlagen der Web-Entwicklung WS 2018/19

Übungsblatt 7

Theorie

Die Lösungen zu den Theorieaufgaben werden in den Tutorien besprochen und im Moodle veröffentlicht. Sie brauchen *keine* Lösungen zu den Theorieaufgaben abgeben!

1. Was ist Node.js?
2. Ist es für Node.js-Webanwendungen notwendig, einen Webserver wie Apache oder nginx einzusetzen?
3. Geben Sie ein Beispiel für eine einfache Node.js-Webanwendung an, die auf dem Übungsserver auf Port 1234 lauscht und beim Aufruf „Hello World“ ausgibt.
4. Wie können Sie eine Node.js-Anwendung starten? Wie erreichen Sie, dass die Anwendung weiterläuft, auch wenn Sie sich per SSH ausgeloggt haben?
5. Erklären Sie den Node.js-Event-Loop in groben Zügen.
6. Erklären Sie den Unterschied zwischen *blocking* und *non-blocking* Funktionsaufrufen in Node.js.
7. Erklären Sie das Prinzip der „*Error-first callbacks*“ in Node.js.
8. Welche Funktion haben die Node.js-Module *http*, *url* und *fs*?
9. Wie können Sie mit Node.js URL-Routing implementieren? Wie können Sie auf HTTP-GET-Parameter zugreifen
10. Was ist npm?
11. Mit welchem Node.js-Modul können Sie mit einer MySQL-Datenbank interagieren?
12. Wie können Sie ein eigenes Node.js-Modul schreiben?
13. Diskutieren Sie positive und negative Seiten von Node.js

Praktische Übungen

Aufgabe 1: *Node.js auf dem Übungsserver „installieren“ (ohne Bewertung) (0 Punkte)*

Laden Sie von `https://nodejs.org/en/download/` die aktuellen Node.js-LTS-Binaries auf den Übungsserver (Linux Binaries (x64)). Entpacken Sie das Archiv in Ihrem Heimatverzeichnis.

Erstellen Sie anschließend in Ihrem Heimatverzeichnis den Ordner *uebung7/hello-world-server*. Erstellen Sie darin eine Datei *index.js* mit einem Hello-World-Server wie im Theorieteil. Der Hello-World-Server steht auch im Moodle zum Download bereit. Sie müssen nur noch den Port ändern. Setzen Sie diesen auf Apache-Port + 2 .

Im *bin*-Ordner Ihrer Node.js-Installation finden Sie das Programm `node`. Starten Sie hiermit im Verzeichnis *uebung7/hello-world-server* Ihren Hello-World-Server, indem Sie `/path/to/node index.js` aufrufen. Hierbei steht `path/to/node` für den vollständigen Pfad zu `node`.

Sie sollten nun im Browser die URL `http://134.2.2.38:<Ihr-Nodejs-Port>` aufrufen können und als Ausgabe „Hello World“ erhalten.

Achtung: In den folgenden Aufgaben werden die Programme, bzw. Befehle `node` und `npm` verwendet. Die Programme finden Sie im *bin*-Ordner Ihrer Node.js-Installation! Sie müssen immer den absoluten oder relativen Pfad zu `node` bzw. `npm` angeben, um die Programme ausführen zu können.

Für die Abgabe: Sie starten Ihre Node.js-Webanwendungen immer nur zum Testen und beenden diese anschließend wieder. D.h. die Webanwendungen sollen nicht dauerhaft laufen! Sie geben in dieser Woche daher keine Links, sondern nur den Quellcode im Moodle ab.

Aufgabe 2: *Node.js-Webserver für statische Dateien (2 Punkte)*

Erstellen Sie einen Node.js-Webserver, der statische Dateien ausliefert. Verwenden Sie hierfür ausschließlich die Module *http*, *url*, *fs* und *path*. Eine „Starthilfe“ für diese Aufgabe finden Sie im Moodle.

Erstellen Sie in Ihrem Heimatverzeichnis den Ordner *uebung7/static-server* und darin den Ordner *htdocs*. Der Ordner *htdocs* enthält später die Dateien, die Ihr Node.js-Webserver ausliefern soll.

Erstellen Sie in *uebung7/static-server* eine Datei namens *index.js* und implementieren Sie darin den Node.js-Webserver mit der folgenden Funktionalität:

- Der Server lauscht auf Ihrem Apache-Port + 2.
- Bei einem eingehenden Request liest der Server den angeforderten URL-Pfad aus, bildet diesen relativ zum *htdocs*-Verzeichnis ab und versucht die entsprechende Datei zu lesen und an den Client zu senden.

  Beispiel: Request an `http://134.2.2.38:12345/test/test.html`. Der URL-Pfad lautet `/test/test.html`. Der Server versucht daher die Datei *htdocs/test/test.html* zu lesen und an den Client zu senden.
- Wenn ein Request an ein Verzeichnis adressiert ist und dieses Verzeichnis existiert, versucht der Server, die Datei *index.html* in diesem Verzeichnis zu lesen und an den Client zu senden.

  Beispiel: Request an `http://134.2.2.38:12345/test`. Angenommen, der Ordner *test* existiert. Der Server versucht, die Datei *htdocs/test/index.html* zu lesen und an den Client zu senden.
- Wird bei einem Request eine nicht existierende Datei angefragt, beantwortet der Server den Request mit HTTP-Statuscode 404 und der Meldung „File not found“.
- Tritt bei der Verarbeitung eines Requests ein Fehler auf, beantwortet der Server den Request mit HTTP-Statuscode 500 und der Meldung „Internal Server Error“.
- Wenn die angefragt Datei existiert, liest der Server diese und sendet sie an den Client. Der HTTP-Statuscode wird hierbei auf 200 gesetzt, der Content-Type wird abhängig von der Dateiendung der angefragten Datei gesetzt. Eine Funktion hierfür steht in der Starthilfe im Moodle bereit.
- Testen Sie Ihren Server, indem Sie im htdocs-Ordner mindestens eine *index.html*-Datei sowie ein Bild namens *img.png* platzieren.
- *Tipp*: Verwenden Sie zum Arbeiten mit Dateien das Modul *fs*. Die Methoden `fs.stat(...)`, `fs.readFile(...)` und `stats.isDirectory()` sind nützlich.

Aufgabe 3: *npm-Modul erstellen (1 Punkt)*

Erstellen Sie ein npm-Modul, mit dem Sie Einträge aus der Tabelle *pages* Ihrer Datenbank (siehe Übung 4) abfragen können[1]. Eine „Starthilfe“ für diese Aufgabe finden Sie im Moodle.

Gehen Sie wie folgt vor:

1. Erstellen Sie in Ihrem Heimatverzeichnis den Ordner *uebung7/webdev-db*.
2. Wechseln Sie in den soeben erstellten Ordner
3. Führen den Befehl `npm init -y` aus. Dadurch wird eine *package.json*-Datei in diesem Verzeichnis erstellt
4. Installieren Sie das npm-Modul *mysql*. Geben Sie hierfür `npm install mysql`.
5. Erstellen Sie die Datei *index.js*

Die Datei *index.js* enthält den Programmcode Ihres Moduls. Das Modul exportiert zwei Funktionen:

- `getAllPages(callback)`: Fragt über das mysql-Modul alle Einträge aus der Tabelle *pages* ab und übergibt diese als zweites Argument an die Callback-Funktion `callback`. Im Fehlerfall wird der Fehler („Error“) als erstes Argument an `callback` übergeben (tritt kein Fehler auf, wird als erstes Argument `null` übergeben!)
- `getPageBySlug(slug, callback)`: Fragt über das mysql-Modul den Eintrag aus der Tabelle *pages* ab, dessen Slug mit `slug` übereinstimmt und übergibt diesen als zweites Argument an die Callback-Funktion `callback` (wenn kein entsprechender Eintrag existiert, wird ein leerer Eintrag übergeben). Im Fehlerfall wird der Fehler („Error“) als erstes Argument an `callback` übergeben (tritt kein Fehler auf, wird als erstes Argument `null` übergeben!).

Bauen Sie jeweils zu Beginn der Funktionen eine Verbindung zur Datenbank auf und beenden Sie diese Verbindung nach abgeschlossener Query bzw. im Fehlerfall. Performance-Optimierung (Connection-Pool und dergleichen) sind *nicht* notwendig!

[1] Wenn Sie Übung 4 nicht bearbeitet haben, führen Sie zunächst das dort beschriebene Setup für die Datenbank aus!

Aufgabe 4: *Node.js-API-Server (2 Punkte)*

Erstellen Sie einen Node.js-API-Server, der mit Hilfe Ihres zuvor erstellten *webdev-db*-Moduls[2] Einträge aus der Tabelle *pages* abruft und JSON-codiert an den Client übergibt.

Gehen Sie wie folgt vor:

1. Erstellen Sie in Ihrem Heimatverzeichnis den Ordner *uebung7/api-server*.
2. Wechseln Sie in den soeben erstellten Ordner
3. Führen den Befehl `npm init -y` aus.
4. Installieren Sie das npm-Modul *webdev-db* mittels `npm install ~/uebung7/webdev-db` (`~/uebung7/webdev-db` ist der Pfad zu Ihrem zuvor erstellten Modul!)
5. Erstellen Sie die Datei *index.js*

index.js implementiert folgende Funktionalität:

- Es wird ein Webserver erstellt, der auf Apache-Port + 2 lauscht.
- Der Content-type der Response wird auf `application/json` gesetzt.
- Bei Requests an die URL `/api/pages` werden mittels des *webdev-db*-Moduls alle Einträge aus *pages* JSON-codiert an den Client geliefert.
- Wenn bei Requests an die URL `/api/pages` zusätzlich ein HTTP-GET-Parameter *slug* übermittelt wird, wird mittels des *webdev-db*-Moduls der Eintrag aus *pages* mit dem entsprechenden Slug JSON-codiert an den Client geliefert.
- Bei erfolgreichen Requests an `/api/pages` wird der HTTP-Statuscode auf 200 gesetzt.
- Wird eine andere URL als `/api/pages` aufgerufen, wird einen JSON-codierte Fehlermeldung an den Client geliefert. Zusätzlich wird der HTTP-Statuscode auf 404 gesetzt.
- Tritt ein Fehler bei der Verarbeitung auf, wird der Fehler JSON-codiert an den Client übermittelt. Zusätzlich wird der HTTP-Statuscode auf 500 gesetzt.

Abgabe bis Montag, 17.12.2018, 12:00 Uhr. Geben Sie im Moodle unter Checkpoint 07 Ihren kompletten Quellcode aus dem Ordner *uebung7* als Archiv ab (als .tar.gz oder .zip).

Sie müssen für diesen Checkpoint keine Links abgeben! Achten Sie vor der Abgabe darauf, dass Ihre Node.js-Anwendungen nicht mehr auf dem Übungsserver ausgeführt werden. Sonst können die Tutoren Ihre Anwendungen nicht ausführen, da die Ports bereits belegt sind.

[2] Verwenden Sie das im Moodle bereitgestellte Modul *webdev-db-dummy* anstatt *webdev-db*, falls Sie die vorherige Aufgabe nicht bearbeitet haben