

Übung 09

1 Koa, Errorhandling und Pug

Erstellen Sie einen Koa-Server der einen Get- und Put-Request an der Adresse *music* entgegen nimmt. Nutzen Sie `routing` um beide Methoden auf die Adresse *music* reagieren zu lassen. Die Get-Methode soll bei Aufruf den Inhalt der beigelegten Datei *musicLib* in der Antwort an den Client zurückschicken. Die Put-Methode soll eine JSON-Datei entgegennehmen und den Inhalt in der beigelegten *musicLib* speichern. Prüfen Sie anschließend ihre Methoden mithilfe von Postman.

Lösungsweg

Zur Bearbeitung der oben gestellten Aufgabe beginnen wir mit der Einrichtung unseres Servers. Hierzu nutzen wir den beigelegten Koa-Server. Schauen wir uns zuerst die benötigten packages an:

```
const Koa = require('koa');
const Router = require('koa-router');
const bodyParser = require('koa-bodyparser');
const fs = require('fs');
```

Zeile für Zeile:

- Koa: Benötigt für den Koa-Server
- Router: Benötigt zum erstellen von Routen zur Bearbeitung von Client-Requests
- bodyParser: Benötigt zum auslesen von Request.body Elementen
- fs: Kurz für File System, benötigt für das lesen und schreiben von Dateien

Als nächstes initialisieren wir unseren Server und unseren Router:

```
const app = new Koa();
var routerMusic = new Router({
  prefix: '/music'
});
```

Durch den Zusatz von `prefix` reagieren alle Methoden des `routerMusic` direkt auf `music` ohne dass diese in der Methodenerstellung zusätzlich definiert werden müssen. Erstellen wir nun die Get-Methode, diese soll auf Anfrage den Inhalt der Datei `musicLib.json` an den Client zurückschicken. Hierfür erstellen wir über unseren Router eine Get-Methode:

```
routerMusic
  .get('/', ctx =>{
    ctx.response.body = fs.readFileSync('musicLib.json', 'utf8');
  })
```

In der Get-Methode wird in den `response.body`, über das File System, die Datei `musicLib.json` eingelesen und direkt als Antwort an den Client geschickt. Anschließend deklarieren wir (In der selben Zeile, also ohne Semikolon) die Put-Methode:

```
.put('/', ctx =>{
  let musicLib = JSON.parse(fs.readFileSync('musicLib.json', 'utf8'));
  musicLib.items.push(ctx.request.body);
  fs.writeFileSync('musicLib.json', JSON.stringify(musicLib));
5   ctx.response.status = 202;
});
```

Innerhalb der Put-Methode, lesen wir zuerst den aktuellen Stand der `musicLib.json` aus und speichern diesen in einer Variabel. Zeitgleich übersetzen wir die ausgelesene Datei, innerhalb des Programmablaufs in ein JSON-Format, um diese anschließend leichter bearbeiten zu können. Dann fügen wir der Liste an Items, im Fall der `musicLib` den Liedern, das vom Request übergebene Element über die `push`-Methode hinzu. Zuletzt übersetzen wir die modifizierte Datei zurück in einen String und schreiben diesen über das File System wieder in die `musicLib.json` und setzen den `response.status` auf 202.

Damit die implementierten Methoden auch funktionieren erstellen wir noch folgende Servereinstellungen:

```
app.use(routerMusic.allowedMethods())
  .use(bodyParser())
  .use(routerMusic.routes());
5 app.listen(8080, () => {
  console.log("Server running on port 8080");
});
```

2 Aufgaben

Bearbeiten Sie nun die folgenden Aufgaben.

Aufgabe 1

Implementieren Sie nun ein Errorhandling innerhalb ihrer Put-Methode. Prüfen Sie, ob die vom Put-Request übermittelten Daten, einen Titel und Artist enthalten. Sollte eines von beiden Fehlen, soll der Server einen Fehler melden mit einem Status 400 und dem Client eine Meldung als Antwort zurückschicken. Analog, sollten beiden Attribute fehlen, soll der Server einen Fehler mit dem Statuscode 500 melden.

Aufgabe 2

Erstellen Sie eine Reihe an Pug-Html-Templates welche angezeigt werden, wenn ein Get-Request an die Adresse `music` oder `film` gesendet wird. Die aus den Pug-Dateien erstellten HTML-Webseiten sollen hierbei den Inhalt von `musicLib`, wenn ein Get-Request an `music` gesendet wird, oder von `filmLib`, bei einem Get-Request an `film`, untereinander auflisten. Jeder Eintrag soll hierbei ein eigenes `div`-Element erhalten. Hierbei sollen sich die Hintergrundfarbe und Randfarbe der `div`-Elemente zwischen den `music` und `film` unterscheiden. Nutzen Sie für Musik und Filme unterschiedliche `pug-mixins` mit denen die Webseite mit Einträgen erweitert werden kann.

Implementieren Sie jeweils eine Get-Methode für `film` und `music`, die auf Anfrage das jeweilige HTML-Dokument erstellt. Sie können hierfür die bereits in der Präsenzaufgabe erstellte Get-Methode für `music` nutzen.

Lösungen zu den Aufgabes

Lösung 2

Listing 1

```
const Koa = require('koa');
const Router = require('koa-router');
const bodyParser = require('koa-bodyparser');
const path = require('path');
5 const fs = require('fs');
var Pug = require('koa-pug');

const app = new Koa();
var router = new Router();
10 var routerMusic = new Router({
  prefix: '/music'
});

15 routerMusic
  .get('/', async ctx =>{
    //ctx.response.body = fs.readFileSync('musicLib.json', 'utf8'); //Teil der
    ↪ Präzensaufgabe
    //Teil von Aufgabe 2
    let musicLib = JSON.parse(fs.readFileSync('musicLib.json', 'utf8'));
20    await ctx.render('index', musicLib);
  })
  .put('/', ctx =>{
    let musicLib = JSON.parse(fs.readFileSync('musicLib.json', 'utf8'));

25    //Teil von Aufgabe 1
    if(typeof ctx.request.body.titel === 'undefined' && typeof
      ↪ ctx.request.body.artist === 'undefined'){
      ctx.throw(500, "Keine Informationen erhalten");
    }

30    if(typeof ctx.request.body.titel === 'undefined' || typeof
      ↪ ctx.request.body.artist === 'undefined'){
      ctx.throw(400, "Informationen nicht vollständig");
    }
    //Teil der Präzensaufgabe
    musicLib.items.push(ctx.request.body);
35    fs.writeFileSync('musicLib.json', JSON.stringify(musicLib));
    ctx.response.status = 202;
  });

//Teil von Aufgabe 1
40 app.on("error", err => {
  console.log("The server encountered an Error: ", err);
});

// Teil von Aufgabe 2
45 const pug = new Pug({
  viewPath: path.resolve(__dirname, 'views' ),
  app: app
});

50 router
```

```

    .get('/filme', async ctx =>{
      let filmLib = JSON.parse(fs.readFileSync('filmLib.json', 'utf8'));
      await ctx.render('index', filmLib);
    });
55
app.use(routerMusic.allowedMethods())
  .use(bodyParser())
  .use(router.allowedMethods())
  .use(router.routes())
60  .use(routerMusic.routes());

app.listen(8080, () => {
65  console.log("Server running on port 8080");
});

```

Listing 2

```

head
  title Meine Film- und Musikbibliothek
  meta(charset="utf-8")
  style.
5    .filmBorder{
      border: 2px solid red;
      background-color: #008;
    }
    .filmFont{
10     text-align: left;
      font-family: sans-serif;
      color: white;
    }
    .musicBorder{
15     border: 2px solid blue;
      background-color: black;
    }
    .musicFont{
20     text-align: left;
      font-family: sans-serif;
      color: white;
    }

```

Listing 3

```

doctype html
html
  include head.pug
  b
5    h1 Meine Film- und Musikbibliothek
    p Schaue dir meine Film- oder Musikbibliothek an
    include itemList.pug

```

Listing 4

```
mixin Film(titel, datum, regie)
    div.filmBorder
        div.filmFont= titel
        div.filmFont= datum
5      div.filmFont= regie

mixin Music(titel, artist)
    div.musicBorder
        div.musicFont= titel
10     div.musicFont= artist
div
    each item in items
        if item.datum
            +Film(item.titel, item.datum, item.regie)
15         if item.artist
            +Music(item.titel, item.artist)
```
