# 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 das 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));
  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());

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 Hintergundfarbe 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.