도서 관리 시스템
박재훈

본 시스템은 따로 라이브러리를 이용하지 않은 Vanilla Node.js를 이용하여 REST API 형태로 개발한 웹 어플리케이션 서버입니다.

파일 구조는 다음과 같습니다.

```
.
├── controller
│   └── book-controller.js
├── data
│   ├── 00001.txt
│   └── 00002.txt
├── model
│   └── book.js
├── package-lock.json
├── package.json
├── server.js
├── service
│   ├── _id.txt
│   └── book-service.js
└── util
    └── util.js
```

따로 데이터베이스를 이용하지 않고 파일 시스템을 통해 데이터를 관리합니다.
데이터들은 data 폴더 내에 텍스트 파일로 관리됩니다.
텍스트 파일의 형식은 아래와 같습니다.

```
1984
조지 오웰
민음사
novel
```

줄(\n)로 나뉘어 관리되며, 각 제목, 저자, 출판사, 장르입니다. (name, author, publisher, genre) 책의 ID는 파일명을 통해 관리됩니다. 가령 ID가 3이라면 파일명은 00003.txt 입니다.
ID는 새로운 책이 등록될 때마다 자동으로 1씩 증가하는 방식으로 부여되며, service 폴더의 _id.txt 파일에 최신 ID를 적어놓는 방식으로 관리됩니다.

server.js에서는 서버를 실행시키고, 들어온 요청을 book-controller.js의 함수들을 통해 처리합니다.
book-controller.js에서는 book-service.js를 통해 데이터(파일)에 대한 CRUD 작업을 수행합니다.
book.js는 모델 정의 파일로, 이 시스템에서 사용하는 Book 모델에 대해 정의합니다.
util.js에는 서버 실행에 관련한 여러 유틸이 정의되어 있습니다.

API spec

GET /api/books

모든 책들에 대한 조회.
(옵션) 정렬을 위한 파라미터 : align, type
align은 정렬의 기준이 되는 필드. (id, name, author, publisher, genre 중 하나)
type은 오름차순 (asc) 혹은 내림차순 (desc) 이며, default는 asc.
파라미터는 쿼리 파라미터 형태로 작성.
모든 책 정보를 배열 형태로 반환.

ex)

GET /api/books	정렬 없이 모든 책 정보 반환
GET /api/books?align=author	저자를 기준으로 오름차순
GET /api/books?align=publisher&type=asc	출판사를 기준으로 오름차순
GET /api/books?align=id&type=desc	ID를 기준으로 내림차순

GET /api/books/:id

특정 책에 대한 조회.
조회를 위한 파라미터. (ID, URL에 작성)
ID에 해당하는 책 정보를 반환하며, 해당하는 책이 없을 시 404 에러 발생.

ex)

GET /api/books/1	1번 책 출력
GET /api/books/mybook	해당하는 책이 없으므로 404 발생

POST /api/books

책 정보 삽입.
책 모델 필드들을 body에 담아 서버에 요청.
요청 예시 :

```
{
   “name”: “sample”,
   “author”: “sample author”,
   “publisher”: “sample publisher”,
   “genre”: “sample genre”
}
```

중복된 제목의 책이 있을 시 409 에러 발생.
존재하지 않는 필드가 있을 경우 에러 발생.
정상적으로 등록 시 등록된 책 객체 정보 반환.

ex)

POST /api/books — sample 책 생성 및 책 정보 반환

```
{
   “name”: “sample”,
   “author”: “sample author”,
   “publisher”: “sample publisher”,
   “genre”: “sample genre”
}
```

POST /api/books — sample 책은 이미 있기 때문에 에러 발생

```
{
   “name”: “sample”,
   “author”: “sample author”,
   “publisher”: “sample publisher”,
   “genre”: “sample genre”
}
```

POST /api/books — publisher, genre 필드가 비어 있기에 에러 발생

```
{
   “name”: “sample2”,
   “author”: “sample author”
}
```

PUT /api/books/:id

책 정보 수정.

URL에 수정할 책의 ID를, body에 수정할 필드명과 값을 담아 요청.

요청 예시 :

/api/books/3

{ “author”: “John Doe” }

3번 책의 저자를 John Doe로 변경.

변경하려는 책 제목이 이미 존재할 경우 변경 불가.

변경 성공 시 변경된 책 객체 정보 반환.

ex)

PUT /api/books/5 — 5번 책의 출판사를 hello로 변경

{ “publisher”: “hello” }

PUT /api/books/mybook — 책이 존재하지 않아 404 에러 발생

{ “name”: “test” }

PUT /api/books/11 — age 필드는 존재하지 않으므로 아무 변화 없음

{ “age”: 20 }

DELETE /api/books/:id

책 정보 삭제.
URL에 삭제할 책의 ID를 담아 요청.
삭제하려는 책이 존재하지 않을 경우 에러 발생.
삭제가 정상적으로 완료되었을 경우 204 (no content) 코드 반환.

ex)

DELETE /api/books/5	5번 책 삭제
DELETE /api/books/mybook	책이 존재하지 않으므로 에러 발생

어플리케이션 작동법

추가 라이브러리가 없기 때문에 Node.js만 설치되어 있으면 실행 가능합니다.
시작 커맨드 : npm start
실행 포트 : 3000 (http://localhost:3000)