

PJT명	도서 데이터를 제공하는 RESTful API 서버 구축	
단계	[Django 개발]	
진행일자	2025.11.14	
예상 구현 시간	필수기능	5H
	추가기능	2H
	심화기능	1H

## 1. 목표

- Django REST Framework를 활용하여 API 서버를 제작할 수 있다.
- HTTP request methods에 대하여 이해한다.
- HTTP response status codes에 대하여 이해한다.
- Many to one relationship(N:1)에 대하여 이해한다.
- Many to many relationship(N:M)에 대하여 이해한다.

## 2. 준비사항

### 1) 제공 파일

- fixture의 형태로 제공되는 초기 데이터 (JSON 파일)
  - books.json, categories.json, threads.json, comments.json

### 2) 개발언어 및 툴

- Python 3.11
- Postman
- Visual Studio Code

### 3) 필수 라이브러리 / 오픈소스

- Django 5.2
- Django REST Framework

### 3. 작업 순서

- 1) 팀원과 같이 요구사항을 확인하고, GitLab에 프로젝트를 생성한다.
  - 프로젝트 이름은 07-pjt로 지정한다.
  - 각 반 담당 강사님을 Maintainer로 설정한다.
- 2) 팀원과 합의하여 협업 방식을 결정하고, 요구사항을 구현한다.
- 3) 작성한 코드들을 정리하고, README를 작성한다.
  - .gitignore 파일을 활용하여 불필요한 파일 및 폴더는 제출하지 않는다.
- 4) README 작성이 완료되면 심화 학습을 진행한다.
- 5) 제출 기한에 맞춰 모든 산출물이 GitLab에 업로드 될 수 있도록 한다.

## 4. 요구사항

AI 기반 도서 분석과 창작 지원 통합 솔루션 서비스를 구축하려고 한다. 다양한 장르와 형태의 도서 데이터를 수집 및 관리하고, 이를 기반으로 AI 기반 도서 분석 기능을 설계 및 구현한다. 또한 작가들에게 아이디어 발상 지원, 플롯 구성 제안, 문장 교정 및 스타일 개선 등 창작 전반에 걸친 다양한 AI 기반 창작 지원 기능을 제공한다. 또한 도서에 대한 사용자 리뷰 및 감상평 공유 커뮤니티 기능을 제공하여, 사용자들이 활발하게 소통하고 정보를 교환할 수 있는 기능을 제공한다. 사용자는 자신이 읽은 도서를 평가하고, 다른 사용자의 리뷰를 참고하여 다음 도서를 선택하는데 도움을 받을 수 있다. 나아가, 관심 도서 목록을 맞춤형으로 구성하는 등 다양한 편의 기능을 제공한다. 팀원과 상의하여 아래 요구사항을 만족할 수 있도록 요구 사항 명세서를 작성 및 구현해보자.

도서 관련 데이터를 제공하는 RESTful API 서버를 만들고자 한다. 도서, 카테고리, 게시글, 댓글 데이터의 조회가 가능하며, 사용자가 게시글을 남기고 관리할 수 있는 API를 제공하는 서버를 구현해보자.

- 요구사항 예시(참고용)
  - 아래의 내용을 참고하여 추가적인 아이디어에 대해 요구사항을 추가 또는 수정하여 기능을 구현한다. 단, **필수 기능은 구현해야 하며, 수정할 수 없다.**

번호	분류	요구사항명	요구사항 상세	우선순위
<b>기능적 요구사항</b>				
F01	프로젝트	프로젝트 구성	도서 커뮤니티 서비스의 API 서버 구현을 위한 Django 프로젝트 및 앱 생성	필수
F02	books model	Category 클래스	카테고리 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F03	books model	Book 클래스	도서 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F04	books model	Thread 클래스	게시글 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F05	books model	Comment 클래스	게시글 댓글 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F06	books serializers	Serializer 클래스	사용자의 입력 데이터 검증 및 응답 데이터 형식을 위한 Serializer 클래스를 개별 요구사항에 맞춰서 구현	필수
F07	books view	category_list	전체 카테고리 데이터를 조회하는 view 함수 구현	필수
F08	books view	book_list	전체 도서 데이터를 조회하는 view 함수 구현	필수
F09	books view	book_detail	단일 도서 데이터를 조회하는 view 함수 구현	필수
F10	books view	thread_list	전체 게시글 데이터를 조회하는 view 함수 구현	필수
F11	books view	thread_detail	단일 게시글 데이터를 조회, 수정, 삭제하는 view 함수 구현	필수
F12	books view	create_thread	전달받은 게시글 데이터를 데이터베이스에 저장하는 view 함수 구현	필수
F13	books view	create_comment	전달받은 게시글 댓글 데이터를 데이터베이스에 저장하는 view 함수 구현	필수

F14	books view	comment_detail	단일 게시글 댓글 데이터를 조회, 수정, 삭제하는 view 함수 구현	필수
F15	AI 활용	도서 검색	도서 제목과 작가를 대상으로 검색을 진행하는 기능 구현	도전

#### 비기능적 요구사항

NF01	프로젝트 관리	Git 활용	개발자 간 작업 충돌이 일어나지 않게끔 Git을 활용하여 프로젝트 관리	필수
NF02	설계	RESTful 원칙	RESTful 설계 원칙을 잘 따르도록 하여 사용성 증대	필수
NF03	보안	HTTP Method 허용	허용된 HTTP Method를 사용하는 요청만 허락하도록 구현	필수

## 1) 프로젝트 관리 (필수)

Git의 기능을 적극적으로 활용하여, 여러 개발자가 동시에 개발할 수 있는 환경을 조성한다.

- 요구사항 번호: NF01
- 팀원과 합의하여 branch 생성 원칙 정리하기
  - Gitflow, GitHub Flow 등을 참고하여 원칙을 정할 수 있음
  - branch 운영 중 문제상황이 발생할 경우 원인에 대한 분석을 꼭 진행할 것
- 팀원과 합의하여 commit 내역 기록 원칙 정리하기
  - 기능 개발, 버그 수정 등 그 목적이 명확히 드러날 수 있도록 commit 메시지를 작성할 것
  - 어느 시점에 commit을 남길지에 대하여 충분한 합의를 이를 것
- 프로젝트 종료 시 초기에 세운 원칙이 잘 지켜졌는지 점검
  - README에 초기에 세운 원칙을 설명하여 정리할 것
  - 생성했던 branch 들에 대한 용도를 정리할 것
  - 작성된 commit들을 GitLab에서 확인하여 스크린샷으로 README에 포함시킬 것

## 2) 프로젝트 및 앱 (필수)

도서 데이터를 생성, 조회, 수정, 삭제할 수 있는 API를 제공하는 Django 프로젝트를 만든다. 명시된 요구사항 이외 서비스를 위해 필요하다고 생각되는 기능 등은 자유롭게 구현해도 무관하며, Bootstrap을 활용하여 자유롭게 스타일링 한다.

- 요구사항 번호: F01
- 프로젝트 이름: mypjt
- 앱 이름: books

## A. books

도서 데이터를 관리하기 위한 앱이다. 요구사항으로 Model 클래스를 정의해보자.

- 요구사항 번호: F02, F03, F04, F05, F06, NF02, NF03
- Category 클래스
  - 카테고리 이름을 지정할 필드(name) 1개 지정
- Book 클래스
  - 도서 제목, 설명, isbn, 커버 이미지, 출판사, 출간일, 저자명, 회원 리뷰 평점(title, description, isbn, cover, publisher, pub\_date, author, customer\_review\_rank)을 지정할 필드 8개 지정
  - Category 클래스와 N:1 관계를 맺고 있는, 도서의 카테고리를 나타내는 필드(category) 지정
- Thread 클래스
  - 게시글 제목, 본문, 독서일, 생성 시간, 수정 시간(title, content, reading\_date, created\_at, updated\_at)을 저장할 필드 5개 지정
  - Book 클래스와 N:1 관계를 맺고 있는, 게시글이 작성된 대상 도서를 나타내는 필드(book) 지정
- Comment 클래스
  - 댓글 내용, 생성 시간, 수정 시간(content, created\_at, updated\_at)을 저장할 필드 3개 지정
  - Thread 클래스와 N:1 관계를 맺고 있는, 댓글이 작성된 게시글을 나타내는 필드(thread) 지정
- Model 클래스를 만든 후 제공된 fixture로 초기 데이터 불러오기
  - 각 Model 클래스의 관계에 주의할 것
- 기능 개발 시 유효성 검증을 위한 Serializer를 요구사항 구현에 필요한 형태로 작성하여 사용
- RESTful 설계 원칙에 알맞게 URL을 설계할 것

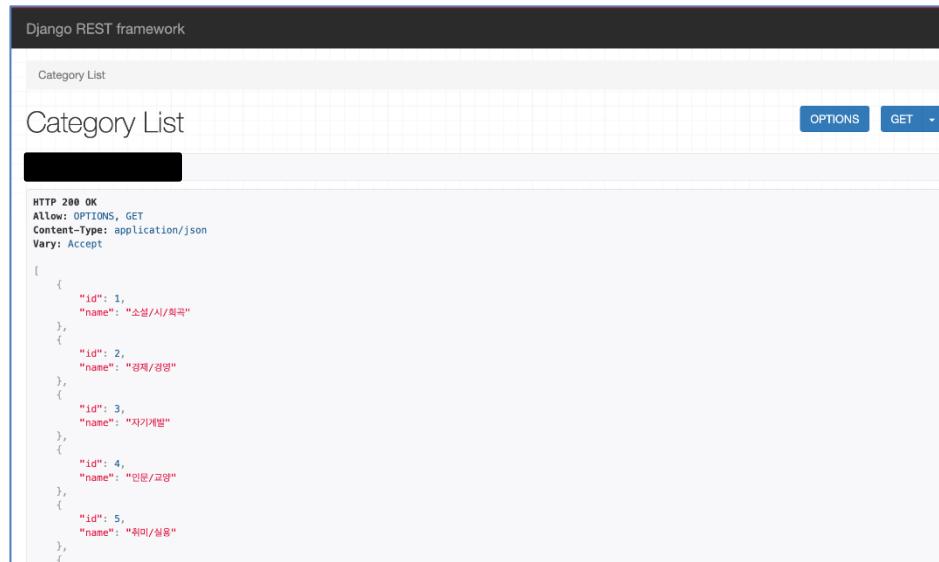
### 3) books 앱 view 함수

도서 서비스와 연관된 다양한 데이터를 생성, 조회, 수정, 삭제하는 기능을 제공하는 books 앱을 구현하자.

#### A. category\_list

전체 도서 카테고리 목록을 제공해주는 category\_list view 함수를 구현한다.

- 요구사항 번호: F07
- 각 카테고리의 id, 이름 데이터가 조회되어야 함
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



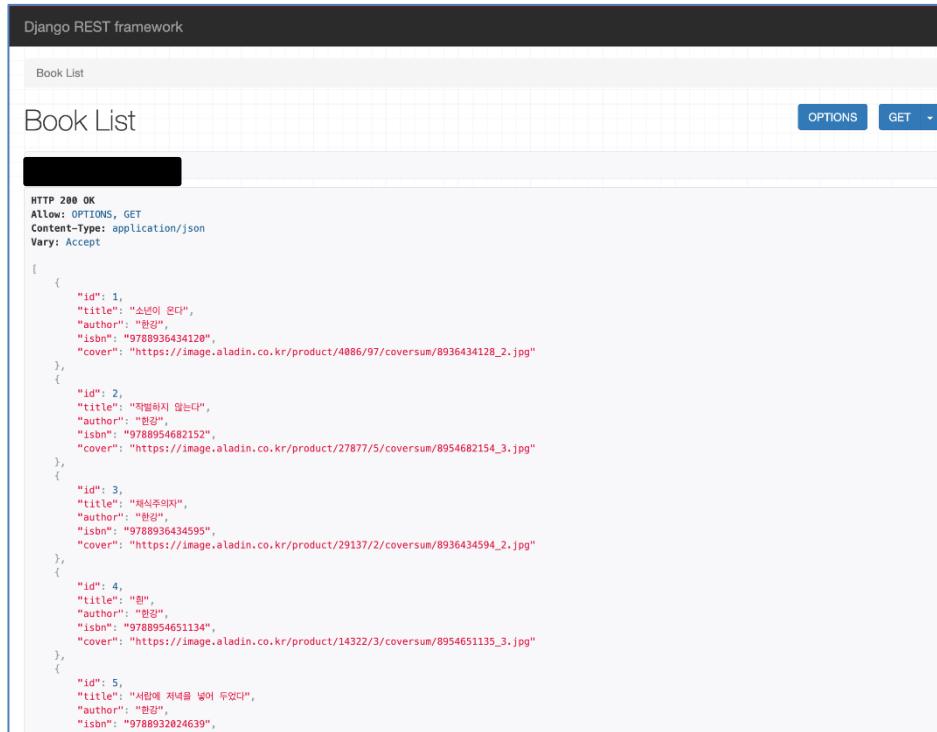
The screenshot shows a Django REST framework browser interface. The title bar says "Django REST framework". Below it, a header bar has "Category List" on the left and "OPTIONS" and "GET" buttons on the right. The main area is titled "Category List". It displays a JSON response with the following data:

```
HTTP 200 OK
Allow: OPTIONS, GET
Content-Type: application/json
Vary: Accept
[
    {
        "id": 1,
        "name": "소설/시/희곡"
    },
    {
        "id": 2,
        "name": "경제/경영"
    },
    {
        "id": 3,
        "name": "자기계발"
    },
    {
        "id": 4,
        "name": "인문/교양"
    },
    {
        "id": 5,
        "name": "취미/실용"
    }
]
```

## B. book\_list

전체 도서 목록을 제공해주는 book\_list view 함수를 구현한다.

- 요구사항 번호: F08
- 각 도서의 id, 제목, 작가, isbn, 커버 이미지 데이터가 조회되어야 함.
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



```
Django REST framework
Book List
OPTIONS GET

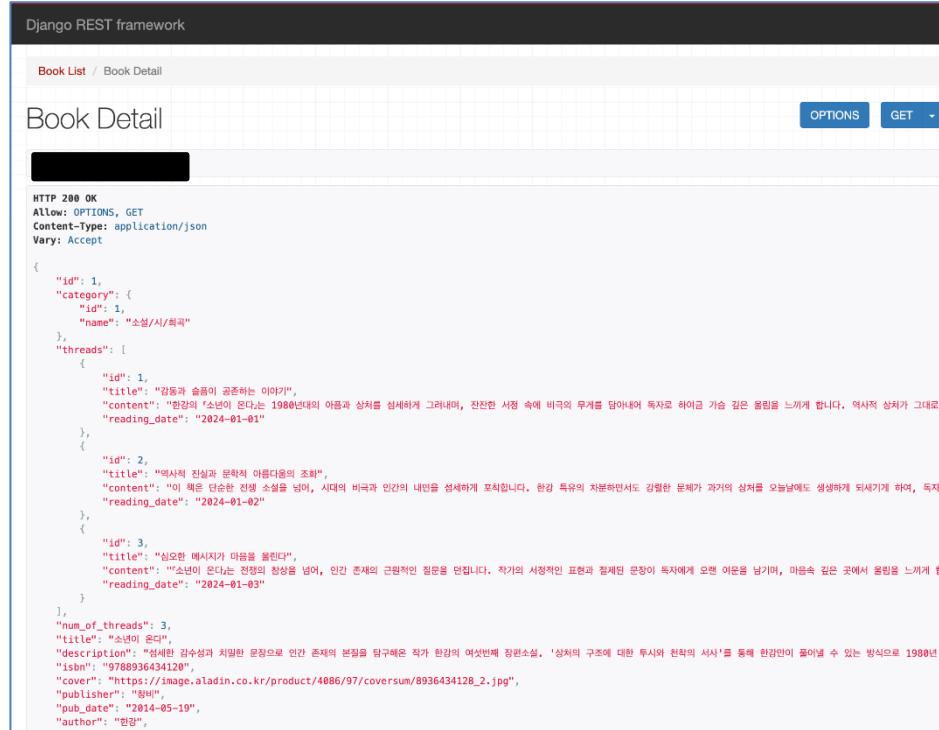
HTTP 200 OK
Allow: OPTIONS, GET
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "title": "소년의 온다",
        "author": "한길",
        "isbn": "9788936434128",
        "cover": "https://image.aladin.co.kr/product/4086/97/coversum/8936434128_2.jpg"
    },
    {
        "id": 2,
        "title": "작별하지 않는다",
        "author": "한길",
        "isbn": "9788954682152",
        "cover": "https://image.aladin.co.kr/product/27877/5/coversum/8954682154_3.jpg"
    },
    {
        "id": 3,
        "title": "제식주의자",
        "author": "한길",
        "isbn": "9788936434595",
        "cover": "https://image.aladin.co.kr/product/29137/2/coversum/8936434594_2.jpg"
    },
    {
        "id": 4,
        "title": "한",
        "author": "한길",
        "isbn": "9788954651134",
        "cover": "https://image.aladin.co.kr/product/14322/3/coversum/8954651135_3.jpg"
    },
    {
        "id": 5,
        "title": "사방에 저녁을 넣어 두었다",
        "author": "한길",
        "isbn": "9788932024639",
        ...
    }
]
```

## C. book\_detail

단일 도서 데이터를 제공해주는 book\_detail view 함수를 구현한다.

- 요구사항 번호: F09
- 대상 도서의 데이터와 함께, 도서의 카테고리, 게시글 목록, 총 게시글 개수 정보가 함께 조회되어야 함
  - 조회되는 게시글의 경우 id, 제목, 내용, 독서일 데이터가 조회되어야 함
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



The screenshot shows a browser window with the title "Django REST framework". Below it, the URL "Book List / Book Detail" is visible. On the right, there are "OPTIONS" and "GET" buttons. The main content area is titled "Book Detail" and contains a large black redacted box. Below the redacted box, the response is displayed as follows:

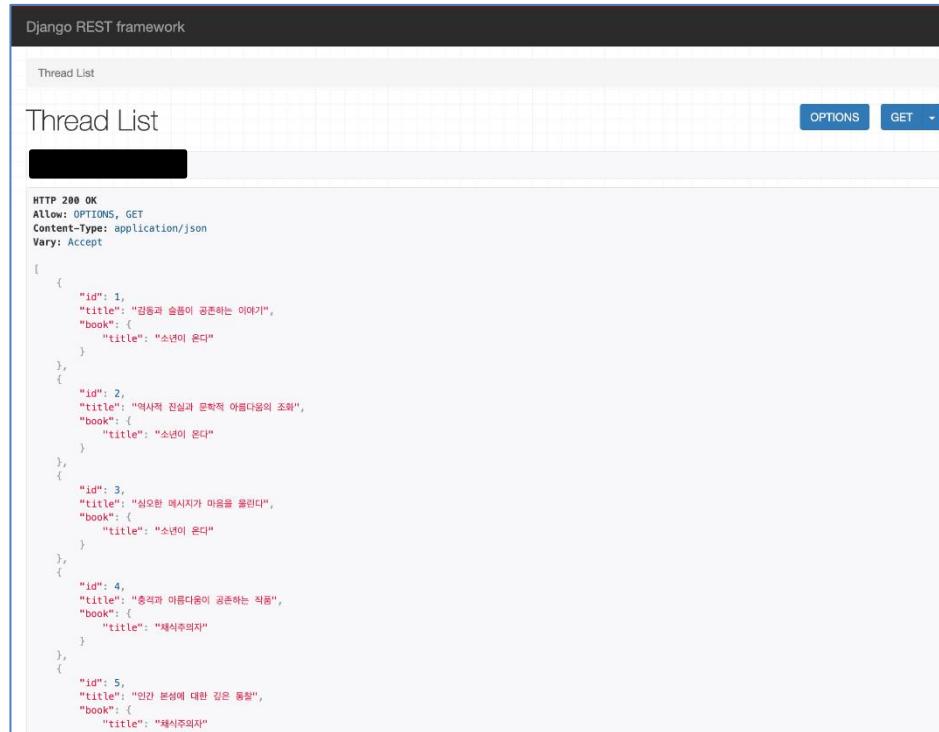
```
HTTP 200 OK
Allow: OPTIONS, GET
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "category": {
        "id": 1,
        "name": "소설/시/희곡"
    },
    "threads": [
        {
            "id": 1,
            "title": "간들과 숨들이 공존하는 이야기",
            "content": "한강의 소년이 온다는 1980년대의 아픔과 상처를 섬세하게 그려내며, 잔잔한 서정 속에 비극의 무게를 담아내어 독자로 하여금 가슴 깊은 울림을 느끼게 합니다. 역사적 상처가 그대로",
            "reading_date": "2024-01-01"
        },
        {
            "id": 2,
            "title": "역사적 진실과 문학적 아름다움의 조화",
            "content": "이 책은 단순한 전쟁 소설을 넘어, 시대의 비극과 인간의 내면을 섬세하게 포착합니다. 한강 특유의 치분하면서도 깊렬한 문제가 과거의 상처를 오늘날에도 생생하게 되새기게 하여, 독자",
            "reading_date": "2024-01-02"
        },
        {
            "id": 3,
            "title": "심오한 메시지가 마음을 불린다",
            "content": "소년이 온다는 전쟁의 침상을 넘어, 인간 존재의 근원적인 질문을 던집니다. 적기의 사정적인 표현과 철제된 문장이 독자에게 오랜 이문을 남기며, 마음속 깊은 곳에서 울림을 느끼게 한",
            "reading_date": "2024-01-03"
        }
    ],
    "num_of_threads": 3,
    "title": "소년이 온다",
    "description": "설레는 감수성과 치밀한 문장으로 인간 존재의 본질을 탐구해온 작가 한강의 여섯번째 장편소설. '상처의 구조에 대한 투시와 천학의 서사'를 통해 한강만이 풀어낼 수 있는 방식으로 1980년",
    "isbn": "9788936434126",
    "cover": "https://image.aladin.co.kr/product/4086/97/coversum/8936434128_2.jpg",
    "publisher": "한강",
    "pub_date": "2014-05-19",
    "author": "한강"
}
```

## D. thread\_list

전체 게시글 목록을 제공해주는 thread\_list view 함수를 구현한다.

- 요구사항 번호: F10
- 각 게시글의 id, 제목과 함께, 대상 도서의 제목이 함께 조회되어야 함
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



The screenshot shows a Django REST framework browser interface titled "Thread List". At the top right, there are buttons for "OPTIONS" and "GET". Below the title, the response content is displayed as a JSON array:

```
[{"id": 1, "title": "김동과 슬범이 공존하는 이야기", "book": {"title": "소년이 온다"}, "book_id": 1}, {"id": 2, "title": "역사적 진실과 문학적 아름다움의 조화", "book": {"title": "소년이 온다"}, "book_id": 2}, {"id": 3, "title": "심오한 메시지가 마음을 울린다", "book": {"title": "소년이 온다"}, "book_id": 3}, {"id": 4, "title": "충직과 이념디중이 공존하는 작품", "book": {"title": "제식주의자"}, "book_id": 4}, {"id": 5, "title": "인간 본성에 대한 깊은 통찰", "book": {"title": "제식주의자"}, "book_id": 5}]
```

## E. thread\_detail

단일 게시글 정보를 request method에 따라 조회, 수정, 삭제해주는 thread\_detail view 함수를 구현한다.

- 요구사항 번호: F11
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method로 요청했을 경우
  - 해당하는 게시글 데이터와 함께, 대상 도서, 댓글 목록, 댓글의 개수 정보가 조회되어야 함
  - 브라우저에서 요청했을 경우 Django REST Framework가 제공하는 HTML을 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- PUT method로 요청했을 경우
  - 유효한 데이터인 경우 대상 게시글 수정
- DELETE method로 요청했을 경우
  - 대상 게시글 삭제
- 브라우저 접근 출력 예시

The screenshot shows a Django REST framework interface for a 'Thread Detail' view. At the top, there's a navigation bar with 'Thread List / Thread Detail'. On the right, there are three buttons: 'DELETE' (red), 'OPTIONS' (blue), and 'GET' (blue). Below the buttons, the page title is 'Thread Detail'. The main content area displays a JSON response:

```
HTTP 200 OK
Allow: OPTIONS, DELETE, PUT, GET
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "book": {
        "title": "소년이 온다"
    },
    "comments": [
        {
            "id": 1,
            "thread": {
                "title": "김동과 술음이 공존하는 이야기"
            },
            "content": "이 강상문은 읽으면서 한강 특유의 사향성과 시대의 아픔이 고스란히 전해져 왔습니다. 눈물과 함께 마음 깊은 곳에서 울림을 느꼈습니다.",
            "created_at": "2024-01-01T12:00:00Z",
            "updated_at": "2024-01-01T12:00:00Z"
        },
        {
            "id": 2,
            "thread": {
                "title": "김동과 술음이 공존하는 이야기"
            },
            "content": "작가의 절제된 문장 하나하나가 당시의 술음과 아픔을 생생하게 전해줘, 오랫동안 잊지 못할 김동을 주았습니다.",
            "created_at": "2024-01-01T12:05:00Z",
            "updated_at": "2024-01-01T12:05:00Z"
        }
    ],
    "num_of_comments": 2,
    "title": "김동과 술음이 공존하는 이야기",
    "content": "『원작의 소년이 된다는 1900년대의 아픔과 상처를 섬세하게 그려내며, 진잔한 서정 속에 비극의 무게를 담아내어 독자로 하여금 가슴 깊은 울림을 느끼게 합니다. 역사적 상처가 그대로 전해지는 듯한 느낌입니다.』",
    "reading_date": "2024-01-01",
    "created_at": "2024-01-01T10:00:00Z",
    "updated_at": "2024-01-01T10:00:00Z"
}
```

## ● Postman 요청 예시

### - PUT method

The screenshot shows a Postman request for a PUT method. The 'Body' tab is selected, showing form-data fields: 'title' (김동과 춤추어 산촌), 'content' (한강의 '소년이' 온다는 1980년대의 이름과 상처를 성세아기 그려내며, 전잔한 서정 ...), and 'reading\_date' (2025-01-12). The response status is 200 OK, with a response body containing JSON data:

```
1, {
2,   "id": 1,
3,   "content": "이 강남을 편안하게 찾은 듯한 평온의 서정과 시대의 아름다움이 고스란히 전해져 왔습니다. 그들과 함께 마음 깊은 곳에서 흘러온 노랫말입니다.",
4,   "created_at": "2024-01-01T12:00:00Z",
5,   "updated_at": "2024-01-01T12:00:00Z"
6, },
7, [
8,   {
9,     "id": 2,
10,    "thread": 1,
11,    "title": "김동과 춤추어 산촌"
12,  },
13,  {
14,    "content": "오늘은 절대로 문장 하나하나가 당시의 습관과 어려움을 생생하게 전파하고, 오랫동안 잊지 못할 감동을 주었습니다.",
15,    "created_at": "2024-01-01T12:05:00Z",
16,    "updated_at": "2024-01-01T12:05:00Z"
17,  }
18, ],
19, {
20,   "run_of_comments": 2,
21,   "title": "김동과 춤추어 산촌",
22,   "content": "한강의 '소년이' 온다는 1980년대의 이름과 상처를 성세아기 그려내며, 전잔한 서정 속에 비극의 부개와 경애로 이어지는 가슴 깊은 울림을 느끼게 합니다. 역사적 상처가 그대로 전해지는 듯한 우정이 인상적입니다",
23,   "reading_date": "2025-01-12",
24,   "created_at": "2024-01-01T10:00:00Z",
25,   "updated_at": "2025-02-27T02:22:14.294045Z"
26, }
```

### - DELETE method

The screenshot shows a Postman request for a DELETE method. The 'Body' tab is selected, showing that the request does not have a body. The response status is 204 No Content, with a response body containing JSON data:

```
1, {
2,   "message": "thread 1 is deleted."
3, }
```

## F. create\_thread

게시글 데이터를 전달받아서 저장하는 create\_thread view 함수를 구현한다.

- 요구사항 번호: F12
- URL은 RESTful 설계 원칙을 고려하여 작성
- POST method에 대해서만 정상 도작해야 함
  - 제목, 내용, 독서일 데이터를 전달받아 새로운 리뷰 생성
  - 유효한 데이터인 경우에만 정상 작동
- 그 외의 method의 경우 405 응답을 반환해야 함
  - 브라우저에서 접근할 경우 데이터를 입력할 수 있는 UI를 표시할 수 있음
- Postman 요청 예시

The screenshot shows a Postman interface with a successful POST request. The request details are as follows:

- Method:** POST
- URL:** [REDACTED]
- Body:** form-data (selected)
- Params:** none
- Headers:** (empty)
- Body (form-data):**

Key	Value	Description
title	역시 노벨상 수상자기	
content	작품 속 성세들은 그 누구도 따라갈 수 없다...	
reading_date	2025-02-12	
- Response Headers:** 201 Created, 14 ms, 597 B
- Response Body (Pretty JSON):**

```
1 {  
2     "id": 10,  
3     "book": {  
4         "title": "소년의 운다"  
5     },  
6     "title": "역시 노벨상 수상자기",  
7     "content": "작품 속 성세들은 그 누구도 따라갈 수 없다...",  
8     "reading_date": "2025-02-12T22:28:01.077400Z",  
9     "created_at": "2025-02-27T02:28:01.077400Z",  
10    "updated_at": "2025-02-27T02:28:01.0774672Z"  
11 }
```

## G. create\_comment

댓글 데이터를 전달받아서 저장하는 create\_comment view 함수를 구현한다.

- 요구사항 번호: F13
- URL은 RESTful 설계 원칙을 고려하여 작성
- POST method에 대해서만 정상 동작해야 함
  - 내용 데이터를 전달받아 새로운 댓글 생성
  - 대상이 되는 게시글은 RESTful 설계 원칙을 고려하여 판별
  - 유효한 데이터인 경우에만 정상 작동
- 그 외의 method의 경우 405 응답을 반환해야 함
  - 브라우저에서 접근할 경우 데이터를 입력할 수 있는 UI를 표시할 수 있음
- Postman 요청 예시

The screenshot shows a Postman interface with a successful API call. The top section shows the request details: Method: POST, URL: [REDACTED], Headers: Content-Type: application/json, Body: form-data. The body contains a key 'content' with value '공감합니다~'. The bottom section shows the response details: Status: 201 Created, Time: 9 ms, Size: 499 B. The response body is a JSON object:

```
1 {  
2   "id": 19,  
3   "thread": {  
4     "title": "의사 노벨상 수상작가"  
5   },  
6   "content": "공감합니다~",  
7   "created_at": "2025-02-27T02:30:01.889922Z",  
8   "updated_at": "2025-02-27T02:30:01.889978Z"  
9 }
```

## H. comment\_detail

단일 댓글 정보를 request method에 따라 조회, 수정, 삭제해주는 comment\_detail view 함수를 구현한다.

- 요구사항 번호: F14
- URL은 RESTful 설계 원칙을 고려하여 작성
- GET method로 요청했을 경우
  - 해당하는 댓글 데이터와 함께, 대상 게시글의 제목이 함께 조회되어야 함
- PUT method로 요청했을 경우
  - 유효한 데이터인 경우 대상 게시글 수정
- DELETE method로 요청했을 경우
  - 대상 게시글 삭제
- Postman 요청 예시
  - GET method

The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** [REDACTED]
- Headers:** (10)
- Body:** (Empty)
- Query Params:** (Empty)
- Params:** (Empty)
- Authorization:** (Empty)
- Scripts:** (Empty)
- Tests:** (Empty)
- Settings:** (Empty)

Below the interface, the response is displayed:

- Status:** 200 OK
- Time:** 7 ms
- Size:** 508 B
- Save Response** button

The response body is a JSON object:

```
1 {  
2   "id": 19,  
3   "thread": [  
4     {"title": "미시 노벨상 수상작가"}  
5   ],  
6   "content": "고급한-느-~",  
7   "created_at": "2025-02-27T02:30:01.889923Z",  
8   "updated_at": "2025-02-27T02:30:01.889978Z"  
9 }
```

- PUT method

The screenshot shows the Postman interface for a PUT request. The URL field contains a redacted URL. The method dropdown is set to "PUT". The "Body" tab is selected, showing a key-value pair: "content" with the value "공감합니다^^". The "Headers" tab shows 10 headers. The response section shows a 200 OK status with a response time of 8 ms and a body size of 508 B. The response content is a JSON object:

```
1 {  
2   "id": 19,  
3   "thread": {  
4     "title": "여기 노출의 수상작가"  
5   },  
6   "content": "공감합니다^^",  
7   "created_at": "2025-02-27T02:36:01.869922Z",  
8   "updated_at": "2025-02-27T02:32:24.727484Z"  
9 }
```

- DELETE method

The screenshot shows the Postman interface for a DELETE request. The URL field contains a redacted URL. The method dropdown is set to "DELETE". The "Body" tab is selected, showing a note: "This request does not have a body". The "Headers" tab shows 7 headers. The response section shows a 204 No Content status with a response time of 10 ms and a body size of 374 B. The response content is a JSON object:

```
1 {  
2   "message": "comment 19 is deleted."  
3 }
```

#### 4) 도전 과제

생성형 AI 도구를 활용하여 도전과제 요구사항을 해결해보자.

- 코드 생성, 아이디어 구상, 문제 해결 방법 탐색 등 다양한 방식으로 활용할 수 있다.
- 사용할 생성형 AI 서비스는 자유롭게 선택한다.
  - 제공되는 GPT API Key를 활용할 경우, 기능별로 정해진 모델을 활용한다.
- 최종 결과물은 AI 생성 내용을 바탕으로 직접 수정 및 개선하여 적용해본다.

보조 수단으로 활용하되, 능동적인 자세로 학습에 임할 것.

- 최종적인 이해와 적용은 자기 주도적 학습을 통해 이루어지며, 배운 내용을 스스로 기록하고 정리하며 학습 효과를 높일 것.

## A. 도서 검색 기능 구현

찾고 싶은 도서 정보를 검색할 수 있는 기능을 구현해보자.

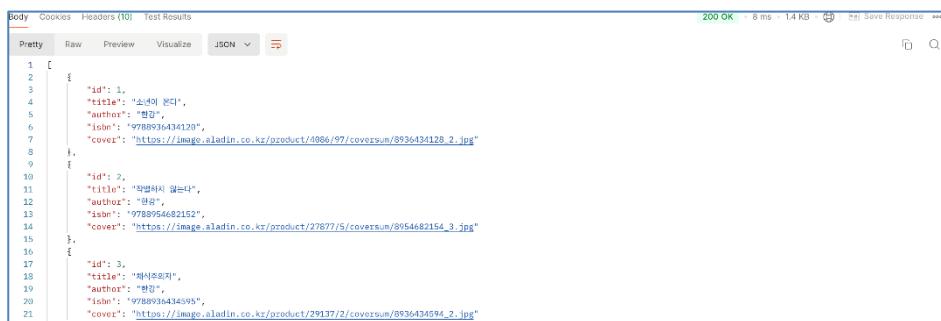
- 요구사항 번호: F15
- 도서 제목과 작가 필드를 대상으로 검색 진행
- 검색을 위한 추가 URL과 view 함수를 작성
  - URL의 각 요소의 목적을 이해하고 활용할 것
- 검색어가 없을 경우 적절한 메시지를 응답할 것
- Postman 응답 예시
  - '데미안'을 검색했을 때 응답



Body Cookies Headers (10) Test Results  
Pretty Raw Preview Visualize JSON ↻ ⚙ Save Response ⚙  
200 OK 17 ms 477 B ⚙ Save Response ⚙  

```
1 [  
2   [  
3     {  
4       "id": 0,  
5       "title": "데미안",  
6       "author": "헤르만 헤세",  
7       "isbn": "9781507011599",  
8       "cover": "https://image.aladin.co.kr/product/9871/8/coverum/k042535558_2.jpg"  
9     }  
10   ]  
11 ]
```

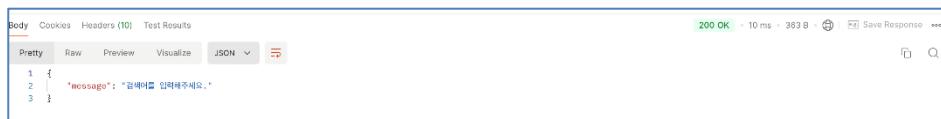
- '한강'을 검색했을 때 응답



Body Cookies Headers (10) Test Results  
Pretty Raw Preview Visualize JSON ↻ ⚙ Save Response ⚙  
200 OK 8 ms 1.4 KB ⚙ Save Response ⚙  

```
1 [  
2   [  
3     {  
4       "id": 1,  
5       "title": "소녀의 본디",  
6       "author": "한강",  
7       "isbn": "97889546134120",  
8       "cover": "https://image.aladin.co.kr/product/1086/97/coverum/8936434128_3.jpg"  
9     },  
10     {  
11       "id": 2,  
12       "title": "작별하지 않는다",  
13       "author": "한강",  
14       "isbn": "9788954682152",  
15       "cover": "https://image.aladin.co.kr/product/27877/5/coverum/8954682154_3.jpg"  
16     },  
17     {  
18       "id": 3,  
19       "title": "화사의회",  
20       "author": "한강",  
21       "isbn": "97889546143095",  
22       "cover": "https://image.aladin.co.kr/product/29137/2/coverum/8936434594_2.jpg"  
23     }  
24   ]  
25 ]
```

- ''을 검색했을 때 응답



Body Cookies Headers (10) Test Results  
Pretty Raw Preview Visualize JSON ↻ ⚙ Save Response ⚙  
200 OK 10 ms 383 B ⚙ Save Response ⚙  

```
1 {  
2   "message": "검색어를 입력해주세요."  
3 }
```

## 5. 참고자료

- Django 5.2  
<https://www.djangoproject.com/start/overview/>
- Django REST Framework  
<https://www.django-rest-framework.org/>

## 6. 결과

제출 기한은 진행일 18시까지이므로 제출 기한을 지킬 수 있도록 한다. 제출은 GitLab을 통해서 이뤄진다.

### ● 산출물과 제출

- 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낀 점을 상세히 기록한 README.md
- 완성된 각 문제 별 소스코드 및 실행 화면 캡쳐본
- 프로젝트 이름은 07-pjt로 지정 및 각자 제출
  - 한 명의 GitLab 계정에 Git 저장소 Push 및 작업
  - 나머지 인원은 제출 시 해당 저장소를 Fork 하여 제출
- 각 반 담당 강사님을 Maintainer로 설정

- 끝 -