**Exercise 2: Online Bookstore - Creating Basic REST Controllers**

Business Scenario:

Implement RESTful endpoints to manage books.

**BookController**

* **Class**: BookController
* **Path**: /books

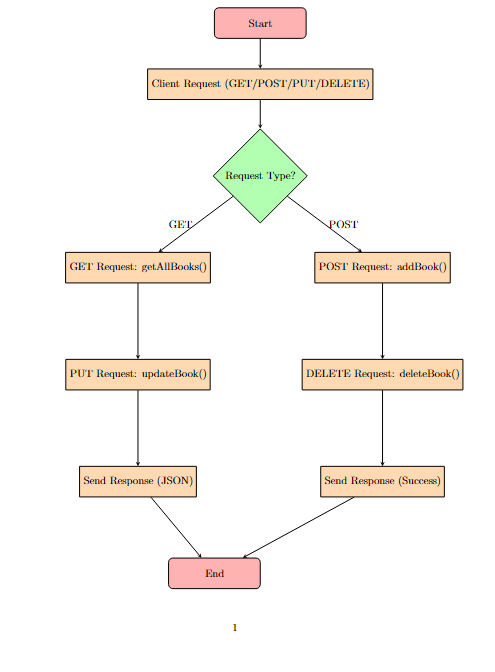
**Endpoints**:

* **GET /books**:
  + **Purpose**: Retrieve a list of all books.
  + **Response**: JSON array of books.
* **POST /books**:
  + **Purpose**: Add a new book.
  + **Request Body**: JSON object of a Book.
  + **Response**: JSON object of the added book.
* **PUT /books/{id}**:
  + **Purpose**: Update the book with the specified ID.
  + **Request Body**: JSON object of updated Book.
  + **Response**: JSON object of the updated book.
* **DELETE /books/{id}**:
  + **Purpose**: Delete the book with the specified ID.
  + **Response**: No content (204 status code).

**2. Book Entity**

* **Attributes**:
  + Long id
  + String title
  + String author
  + Double price
  + String isbn

**FLOWCHART :**

1. 

**Explanation :**

**1.Start:**

* The process begins when the application is running, waiting to handle incoming client requests.

**2. Client Request:**

* This is the initial step where a client sends an HTTP request to the application. The request could be of different types: GET, POST, PUT, or DELETE.

**3. Request Type Decision:**

* At this point, the application checks what type of HTTP request has been received. This decision determines the next step in the flow.

**4. GET Request:**

* **Action**: If the request is a GET request, the application calls the getAllBooks() method in the BookController.
* **Outcome**: This method retrieves a list of all books and prepares the response.

**5. POST Request:**

* **Action**: If the request is a POST request, the application calls the addBook() method.
* **Outcome**: This method adds a new book to the system based on the data provided in the request body and prepares the response with the newly added book.

**6. PUT Request:**

* **Action**: If the request is a PUT request, the application calls the updateBook() method.
* **Outcome**: This method updates the details of an existing book based on the ID provided in the URL and the updated data in the request body. The updated book is then prepared as the response.

**7. DELETE Request:**

* **Action**: If the request is a DELETE request, the application calls the deleteBook() method.
* **Outcome**: This method deletes the book with the specified ID, and the application prepares a success response indicating the deletion.

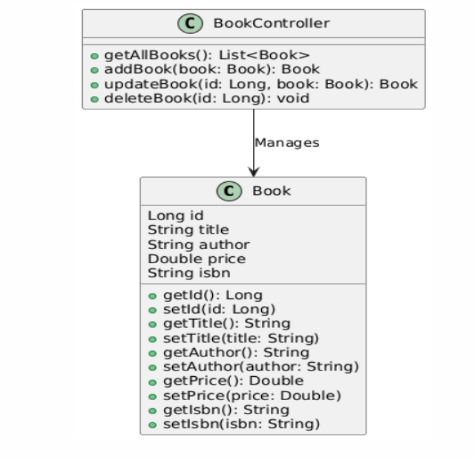
**8. Send Response:**

* After handling the request (whether GET, POST, PUT, or DELETE), the appropriate response is sent back to the client.
  + For GET, POST, and PUT requests, the response typically contains JSON data (e.g., a list of books or the updated book).
  + For DELETE requests, the response is typically a success message or status code.

**9. End:**

* The flow ends after the response is sent back to the client. The application then waits for the next client request.

**CLASS DIAGRAM :**



**Book Class:**

* **Attributes**:
  + Long id: Represents the unique identifier for each book.
  + String title: Represents the title of the book.
  + String author: Represents the author of the book.
  + Double price: Represents the price of the book.
  + String isbn: Represents the ISBN number of the book.
* **Methods**:
  + Getter and setter methods for each attribute (e.g., getId(), setId(Long id), getTitle(), setTitle(String title), etc.).
  + These methods allow accessing and modifying the attributes of a Book object.

**2. BookController Class:**

* **Methods**:
  + getAllBooks(): Returns a list of all Book objects.
  + addBook(Book book): Adds a new Book object and returns the added book.
  + updateBook(Long id, Book book): Updates the book details for a given ID and returns the updated book.
  + deleteBook(Long id): Deletes the Book object with the specified ID.
* This class acts as a controller that manages the CRUD operations for the Book entity. It handles the interactions between the client and the Book objects.

**3. Relationship:**

* The BookController class has a dependency on the Book class, as indicated by the arrow (-->). This relationship represents that the BookController manages Book objects through its methods.