Creating a RESTful API application using Spring Boot MVC. We'll create a simple CRUD API for managing a list of books.

**Step 1: Set Up the Development Environment**

- Install Java Development Kit (JDK) if not already installed.

- Install an Integrated Development Environment (IDE) such as IntelliJ IDEA or Eclipse.

- Create a new Spring Boot project using the Spring Initializr or the IDE's project creation wizard. Make sure to include the **Spring Web and Spring Data JPA dependencies.**

**Step 2: Create a Model Class**

- In your project, create a new package called `com.example.api.model`.

- Create a new class called `Book` in the `com.example.api.model` package.

```java

package com.example.api.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

// Constructor, getters, and setters

}

```

**Step 3: Create a Repository Interface**

- In your project, create a new package called `com.example.api.repository`.

- Create a new interface called `BookRepository` in the `com.example.api.repository` package.

```java

package com.example.api.repository;

import com.example.api.model.Book;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {

}

```

**Step 4: Implement Service Layer**

- In your project, create a new package called `com.example.api.service`.

- Create a new class called `BookService` in the `com.example.api.service` package.

```java

package com.example.api.service;

import com.example.api.model.Book;

import com.example.api.repository.BookRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class BookService {

private final BookRepository bookRepository;

@Autowired

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

public Book getBookById(Long id) {

return bookRepository.findById(id).orElse(null);

}

public Book createBook(Book book) {

return bookRepository.save(book);

}

public Book updateBook(Long id, Book book) {

Book existingBook = bookRepository.findById(id).orElse(null);

if (existingBook != null) {

existingBook.setTitle(book.getTitle());

existingBook.setAuthor(book.getAuthor());

return bookRepository.save(existingBook);

}

return null;

}

public void deleteBook(Long id) {

bookRepository.deleteById(id);

}

}

```

**Step 5: Create Controller Class**

- In your project, create a new package called `com.example.api.controller`.

- Create a new class called `BookController` in the `com.example.api.controller` package.

```java

package com.example.api.controller;

import com.example.api.model.Book;

import com.example.api.service.BookService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/api/books")

public class BookController {

private final BookService bookService;

@Autowired

public BookController(BookService bookService) {

this.bookService = bookService;

}

@GetMapping(“/allbooks”)

public List<Book> getAllBooks() {

return bookService.getAllBooks();

}

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookService.getBookById(id);

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookService.createBook(book);

}

@PutMapping("/{id}")

public Book updateBook(@PathVariable Long id, @RequestBody Book book) {

return bookService.updateBook(id, book);

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

bookService.deleteBook(id);

}

}

```

**Step 6: Configure Application Properties**

- Open the `application.properties` file in your project's `src/main/resources` directory.

- Add the following configuration for the H2 in-memory database:

```

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

```

**Step 7: Build and Run the Application**

- Build the project using Maven or Gradle.

- Run the application using the IDE or command line.

- The Spring Boot application will start, and you can access the API endpoints using the base URL `http://localhost:8080/api/books`.

Now you have a basic RESTful API application for managing books. You can use tools like cURL or Postman to interact with the API endpoints and perform CRUD operations on books.

**Here are some example API requests you can try:**

- Get all books: GET `http://localhost:8080/api/books`

- Get a specific book by ID: GET `http://localhost:8080/api/books/{id}`

- Create a new book: POST `http://localhost:8080/api/books` with a JSON payload containing book details.

- Update an existing book: PUT `http://localhost:8080/api/books/{id}` with a JSON payload containing updated book details.

- Delete a book: DELETE `http://localhost:8080/api/books/{id}`