**Create a Spring Web Project using Maven**

**Spring REST using Spring Boot 3**

Vaishnavi

11/7/2025

In this project, we are creating a basic Spring Boot web application using Maven. The app will demonstrate the standard project structure and usage of annotations like @SpringBootApplication. We'll also explore the project folder structure, pom.xml, and how to verify that the application runs successfully.

**Objective:**

The objective of this project is to create a basic Spring Boot web application using Maven that demonstrates the foundational setup and structure of a Spring project. This includes understanding the standard folder hierarchy, configuring dependencies using pom.xml, exploring key Spring Boot annotations like @SpringBootApplication, and verifying the application startup using IntelliJ IDEA. The project serves as a foundational step for building web-based applications and microservices using the Spring Boot framework.

**Implementation:**

### Generate Project from Spring Initializr

Go to: [https://start.spring.io](https://start.spring.io" \t "_new)

Set:

Group: com.cognizant

Artifact: spring-learn

Add Dependencies:

Spring Web

Spring Boot DevTools.

Click **Generate**, and download the .zip file.

Extract the zip file to a suitable folder.

### Open Project in IntelliJ IDEA

Open IntelliJ IDEA.

Click **File > Open**.

Browse and select the extracted spring-learn folder.

IntelliJ will auto-detect it as a Maven project and build it.

If prompted to import as Maven project, click **Yes** or **Import**.

**SpringlearnApplication.java:**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringlearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringlearnApplication.class, args);

}

}

**Book.java:**

package com.cognizant.springlearn;

import jakarta.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private double price;

public Book() {}

public Book(String title, String author, double price) {

this.title = title;

this.author = author;

this.price = price;

}

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getAuthor() { return author; }

public void setAuthor(String author) { this.author = author; }

public double getPrice() { return price; }

public void setPrice(double price) { this.price = price; }

}

**BookController.java:**

package com.cognizant.springlearn;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

@Controller

public class BookController {

@Autowired

private BookService service;

@GetMapping("/books")

public String viewBooks(Model model) {

model.addAttribute("books", service.getAllBooks());

return "books";

}

@GetMapping("/books/add")

public String showAddForm(Model model) {

model.addAttribute("book", new Book());

return "book-form";

}

@PostMapping("/books/save")

public String save(@ModelAttribute("book") Book book) {

service.saveBook(book);

return "redirect:/books";

}

@GetMapping("/books/edit/{id}")

public String edit(@PathVariable Long id, Model model) {

model.addAttribute("book", service.getBookById(id));

return "book-form";

}

@GetMapping("/books/delete/{id}")

public String delete(@PathVariable Long id) {

service.deleteBook(id);

return "redirect:/books";

}

}

**BookRepository.java:**

package com.cognizant.springlearn;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**BookService.java:**

package com.cognizant.springlearn;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class BookService {

private final BookRepository repository;

@Autowired

public BookService(BookRepository repository) {

this.repository = repository;

}

public List<Book> getAllBooks() {

return repository.findAll();

}

public void saveBook(Book book) {

repository.save(book);

}

public Book getBookById(Long id) {

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

}

public void deleteBook(Long id) {

repository.deleteById(id);

}

}

**BookServiceTest.java:**

package com.cognizant.springlearn;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import java.util.Arrays;

import java.util.List;

import java.util.Optional;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

public class BookServiceTest {

@Test

public void testGetAllBooks() {

BookRepository mockRepo = Mockito.mock(BookRepository.class);

List<Book> mockBooks = Arrays.asList(

new Book("Book A", "Author A", 100.0),

new Book("Book B", "Author B", 200.0)

);

when(mockRepo.findAll()).thenReturn(mockBooks);

BookService service = new BookService(mockRepo);

List<Book> books = service.getAllBooks();

assertEquals(2, books.size());

assertEquals("Book A", books.get(0).getTitle());

}

@Test

public void testSaveBook() {

BookRepository mockRepo = mock(BookRepository.class);

BookService service = new BookService(mockRepo);

Book book = new Book("Test Book", "Tester", 150.0);

service.saveBook(book);

verify(mockRepo, times(1)).save(book);

}

@Test

public void testGetBookById() {

BookRepository mockRepo = mock(BookRepository.class);

Book book = new Book("Java Book", "Oracle", 299.99);

book.setId(1L);

when(mockRepo.findById(1L)).thenReturn(Optional.of(book));

BookService service = new BookService(mockRepo);

Book foundBook = service.getBookById(1L);

assertNotNull(foundBook);

assertEquals("Java Book", foundBook.getTitle());

}

@Test

public void testDeleteBook() {

BookRepository mockRepo = mock(BookRepository.class);

BookService service = new BookService(mockRepo);

service.deleteBook(1L);

verify(mockRepo, times(1)).deleteById(1L);

}

}

**Book.html:**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Add/Edit Book</title>

<link rel="stylesheet" th:href="@{/style.css}" />

</head>

<body>

<div class="container">

<h1 th:text="${book.id == null} ? 'Add Book' : 'Edit Book'"></h1>

<form th:action="@{/books/save}" th:object="${book}" method="post">

<input type="hidden" th:field="\*{id}" />

<label>Title:</label>

<input type="text" th:field="\*{title}" required/><br/>

<label>Author:</label>

<input type="text" th:field="\*{author}" required/><br/>

<label>Price:</label>

<input type="number" step="0.01" th:field="\*{price}" required/><br/>

<button type="submit" class="btn">Save</button>

</form>

</div>

</body>

</html>

**Book-form.html:**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Books</title>

<link rel="stylesheet" th:href="@{/style.css}" />

</head>

<body>

<div class="container">

<h1>Book List</h1>

<a class="btn" href="/books/add">Add New Book</a>

<table>

<tr><th>ID</th><th>Title</th><th>Author</th><th>Price</th><th>Action</th></tr>

<tr th:each="book : ${books}">

<td th:text="${book.id}"></td>

<td th:text="${book.title}"></td>

<td th:text="${book.author}"></td>

<td th:text="${book.price}"></td>

<td>

<a th:href="@{/books/edit/{id}(id=${book.id})}">Edit</a> |

<a th:href="@{/books/delete/{id}(id=${book.id})}">Delete</a>

</td>

</tr>

</table>

</div>

</body>

</html>

**Application.properties:**

spring.application.name=springlearn

server.port=8080

spring.datasource.url=jdbc:mysql://localhost:3306/bookverse

spring.datasource.username=root

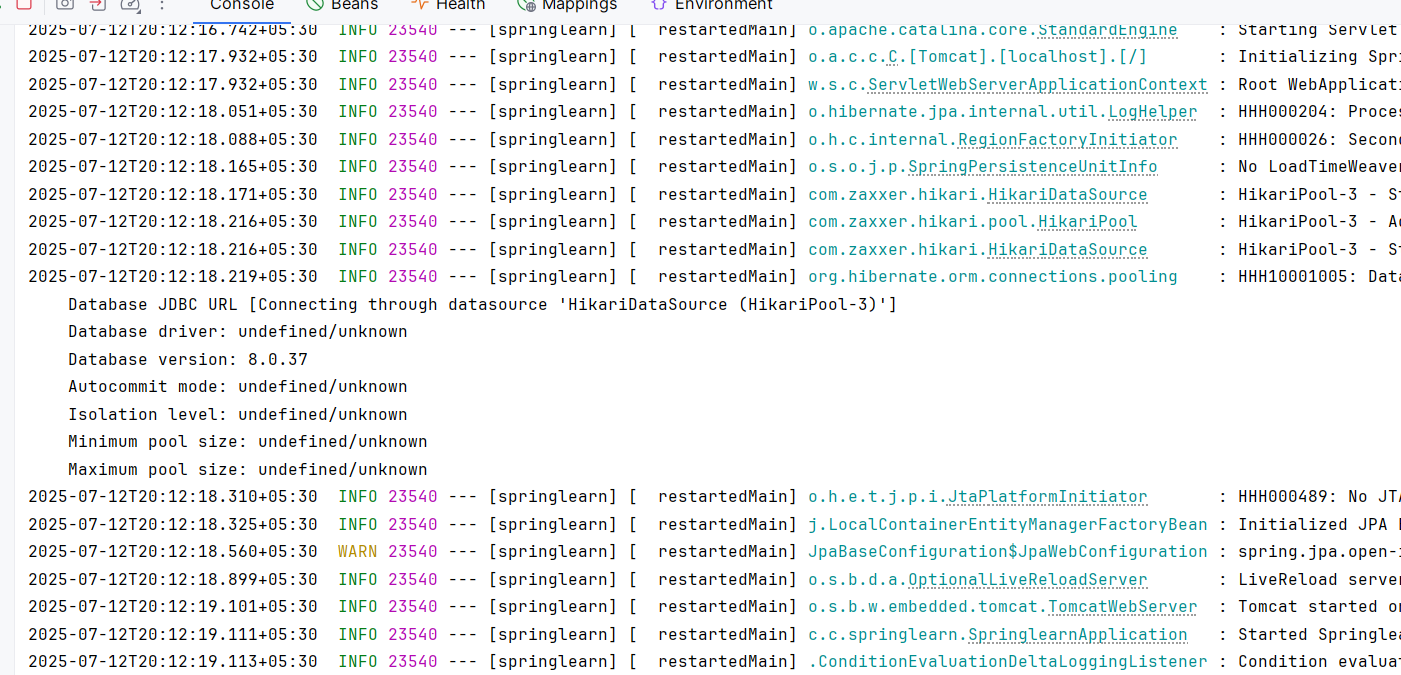
spring.datasource.password=root

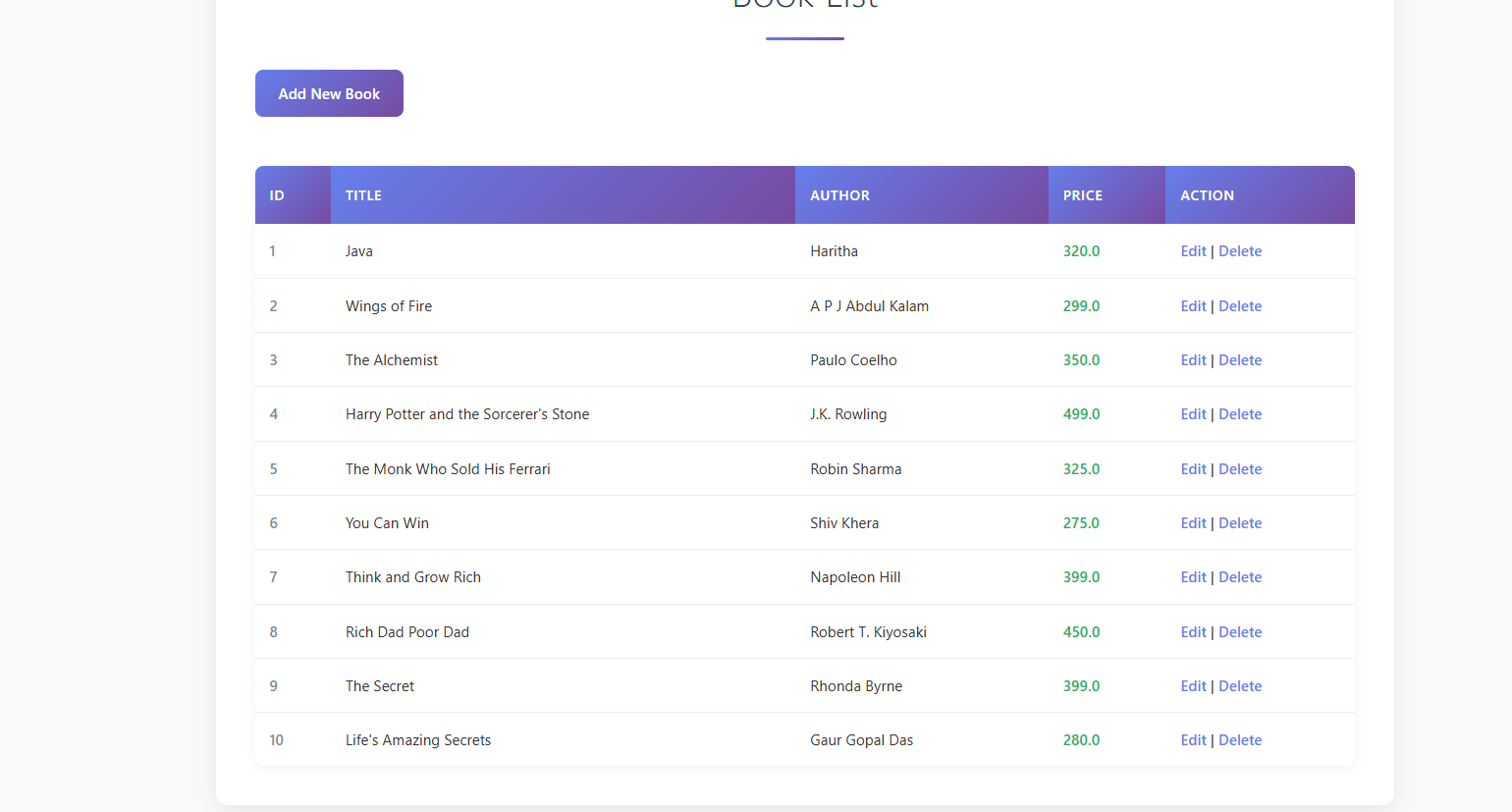
spring.jpa.hibernate.ddl-auto=update

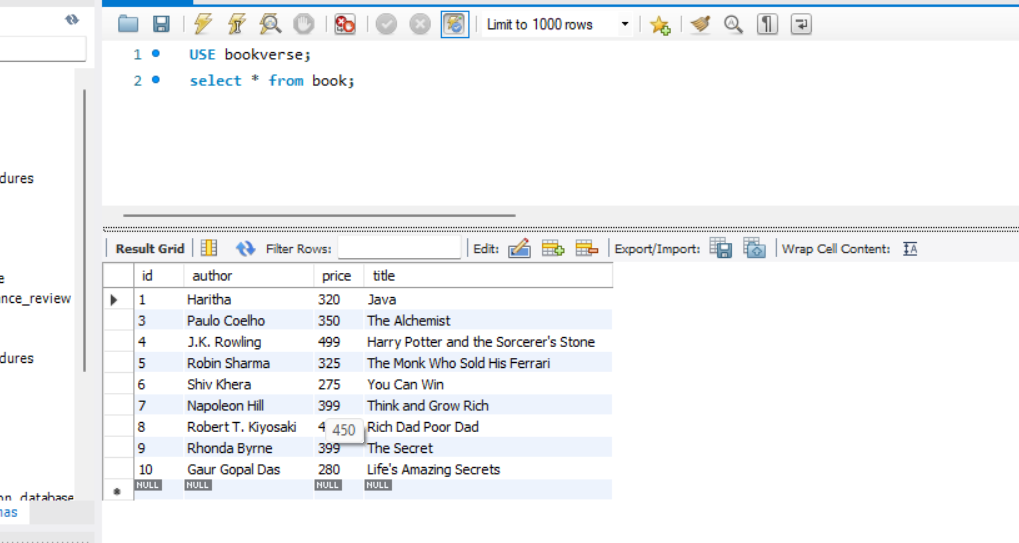
spring.jpa.show-sql=true

spring.thymeleaf.cache=false

**Output:**

****

****

****