**Configuring a Basic Spring Application for Library Management**

This exercise explains how to configure a basic Spring application for managing a library's book collection. Spring is a popular Java framework that simplifies application development by handling dependency injection and object lifecycle management.

**Understanding the Problem**

We need to develop a backend system for a library application. This system will manage book information, including adding, removing, and searching for books. Spring will be used to structure the application and manage the interactions between different components.

**Setting Up a Spring Project (Steps 1.1 & 1.2)**

1. **Creating a Maven Project:**
   * We start by creating a Maven project named "LibraryManagement". Maven is a build automation tool that simplifies managing project dependencies.
2. **Adding Spring Core Dependencies:**
   * The pom.xml file defines the project's dependencies. We add the spring-context dependency from Spring Core to enable Spring functionalities in our application.

**Configuring the Application Context (Step 2)**

* An XML configuration file named applicationContext.xml is created in the src/main/resources directory.
* This file defines beans, which are objects managed by Spring. In this case, we define beans for two classes: BookService and BookRepository. Spring will create instances of these classes and handle their lifecycle.

**Defining Service and Repository Classes (Step 3)**

1. **BookService:**
   * This class represents the service layer responsible for managing book operations in the library. It has a private field for the BookRepository bean and a setter method for dependency injection. The manageBooks method can be used to perform various book-related tasks.
2. **BookRepository:**
   * This class represents the data access layer, responsible for interacting with the data source (like a database) to store and retrieve book information. Here, the saveBook method demonstrates a basic data persistence operation.

**Note:** In a real application, the BookRepository would likely interact with a database using libraries like Spring Data JPA.

**Running the Application (Step 4)**

1. **Creating a Main Class:**
   * A MainApp class is created to bootstrap the Spring application context. It loads the applicationContext.xml file and retrieves the BookService bean using its name (bookService).
2. **Running the Main Class:**
   * Executing the MainApp class triggers the Spring context initialization and injects the BookRepository bean into the BookService. Finally, the manageBooks method of the BookService is called, printing "Managing books in the library..." to the console.

This exercise demonstrates how Spring manages beans and facilitates communication between different parts of the application. While the provided code uses simple functionalities, Spring allows for building complex and scalable enterprise applications.