**Implementing Constructor and Setter Injection**

**Configuring Constructor Injection (Step 1):**

* The applicationContext.xml code demonstrates how to configure constructor injection for BookService.
  + <constructor-arg ref="bookRepository" /> Injects the bookRepository bean reference as an argument during object creation using the constructor.

**Configuring Setter Injection (Step 2):**

* The code shows modifications to the BookService class:
  + A constructor is added with @Autowired to enable constructor injection (optional in this case).
  + The existing setter method with @Autowired remains for setter injection.

**Configuring Setter Injection in XML (Step 2):**

* While the setter is already marked with @Autowired, Spring can still use it for injection, even though the constructor is also configured for injection.
* Including both <constructor-arg> and <property> elements in applicationContext.xml doesn't necessarily mean both will be used. Spring prioritizes constructor injection if a suitable constructor exists.

**Testing the Injection (Step 3):**

* Running LibraryManagementApplication verifies that Spring injects dependencies using constructor injection (if a suitable constructor exists) and doesn't necessarily use the setter defined in XML.

**Summary:**

* Both constructor and setter injection for BookService has implemented.
* Spring prioritizes constructor injection if a suitable constructor is defined with @Autowired.
* Both annotations on the class can be used for flexibility, but Spring will only use one during object creation.

**Extra Notes:**

* Constructor injection is generally preferred because it enforces dependency requirements at object creation and promotes immutability.
* Setter injection can be useful for situations where object state needs to be modified after creation.