Documentation: Spring Bookstore Application

This documentation explains the Spring Bookstore application, which is designed to manage books using the Spring Framework with Java-based configuration.

1. Book Class (`com.springcore.bookstore.Book`):

The `Book` class represents a book entity with the following properties:

- `title` (String): The title of the book.
- `author` (String): The author of the book.
- `price` (String): The price of the book.

The class contains constructors, getters, setters, and an overridden `toString()` method to facilitate object creation and retrieval.

2. BookConfiguration Class (`com.springcore.bookstore.BookConfiguration`):

The `BookConfiguration` class is a Java configuration class marked with the `@Configuration` annotation. It defines three `@Bean` methods to create instances of different books using the `@Value` annotation to set the book properties.

2.1. `javaBook()` method:
- Creates a `Book` instance for a Java book with title "Java book," author "Java author," and price "Java price."

2.2. `pythonBook()` method:
- Creates a `Book` instance for a Python book with title "pythonBook book," author "pythonBook author," and price "pythonBook price."

2.3. `javascriptBook()` method:
- Creates a `Book` instance for a JavaScript book with title "javascriptBook book," author "javascriptBook author," and price "javascriptBook price."

3. MainClass (`com.springcore.bookstore.MainClass`):

The `MainClass` serves as the entry point of the application. It loads the Spring context from the XML file and retrieves beans for three books (Java, Python, and JavaScript) from the context.

4. XML Configuration (`bookstore.xml`):

The XML configuration file is used to enable component scanning and locate the `@Configuration` class (`BookConfiguration`). It includes the following elements:

- `context:component-scan`: Scans the package `com.springcore.bookstore` to discover and register Spring beans defined with annotations.

How the Application Works:

1. The `MainClass` is executed as the entry point of the application.
2. It loads the Spring context using the `ClassPathXmlApplicationContext`, which reads the `bookstore.xml` configuration file.
3. The `BookConfiguration` class is discovered during component scanning and processed as a configuration class.
4. Three `Book` instances (Java, Python, and JavaScript books) are created by calling the respective `@Bean` methods.
5. The books are retrieved using their bean names ("javaBook," "pythonBook," and "javascriptBook") from the Spring context.
6. Each book's details are printed using the overridden `toString()` method.

Running the Application:

To run the Spring Bookstore application, make sure you have Java and Spring dependencies set up properly. Follow these steps:

1. Compile the Java classes using your preferred method (e.g., IDE or command-line).
2. Ensure that the `bookstore.xml` file is available in the classpath (usually in the same directory as compiled classes).
3. Run the `MainClass` to execute the application.
4. Observe the console output to see the details of the three books (Java, Python, and JavaScript) printed.

Conclusion:

The Spring Bookstore application demonstrates how to use Java-based configuration to manage beans in a Spring application. It defines a `Book` class as a simple entity, a `BookConfiguration` class for creating bean instances, and an XML configuration file for component scanning. The application then retrieves and prints the details of three books using Spring's dependency injection and component scanning capabilities.