Spring
More container concepts

Overview

In previous chapters, we used simple string values for property injection in our configuration files. However, in real-world applications, we often need to inject full objects as dependencies.

This chapter covers:

- Injecting object dependencies in Spring.
- Setter Injection vs. Constructor Injection.
- Configuring dependencies in an XML file.

Injecting object dependencies

Consider a class PurchasingServiceImpl, which has two dependencies:

- AccountsService
- BookService

These dependencies are injected via setter methods:

```
public class PurchasingServiceImpl implements PurchasingService {
     private AccountsService accounts;
     private BookService books;

     public void setAccountsService (AccountsService accounts) {
          this.accounts = accounts;}
     public void setBookService (BookService books) {
          this.books= books; }
```

Using Dependencies in Methods

The injected services are used within methods, such as buyBook:

```
@Override
 public void buyBook(String isbn) {
     //To find the book
     Book book = books.getBookByIsbn(isbn);
     //Raise invoice
     accounts.raiseInvoice(book);
 }
```

Notice that the implementation details of AccountsService and BookService are not defined in the class. Instead, they will be configured externally in Spring's XML configuration file.

Defining Beans in XML

We define all beans in the application.xml file:

```
<bean id="accountsService" class="se.yrgo.spring.services.AccountsServiceMockImpl"/>
<bean id="bookService" class="se.yrgo.spring.services.BookServiceMockImpl"/>
<bean id="purchasingService" class="se.yrgo.spring.services.PurchasingServiceImpl"/>
```

However, this configuration is not complete yet!

If we run the code, we will encounter a NullPointerException because we have not injected the dependencies into PurchasingServiceImpl.

Completing Dependency Injection

To fully wire our dependencies, we need to reference them in the XML configuration:

```
<bean id="purchasingService" class=
"se.yrgo.spring.services.PurchasingServiceImpl">
    <property name = "bookService" ref="bookService" />
    <property name= "accountsService" ref="accountsService" />
</bean>
```

Important Notes

- The property name in XML should match the variable names in PurchasingServiceImpl.
- We use ref instead of value, since we are injecting objects (not primitive values or strings).

Why ref and not value?

value is used for primitive values (e.g., Strings, integers).

ref is used for injecting objects that are defined elsewhere in the Spring container.

Constructor Injection

An alternative to setter injection is constructor injection, where dependencies are passed via the constructor instead of setter methods.

Example: Constructor Injection

```
public class PurchasingServiceImpl implements PurchasingService {
    private final AccountsService accounts;
    private final BookService books;

    public PurchasingServiceImpl(AccountsService accounts, BookService books) {
        this.accounts = accounts;
        this.books = books;
    }
}
```

Configuring Constructor Injection in XML

```
<bean id="purchasingService" class= "se.yrgo.spring.services.PurchasingServiceImpl">
        <constructor-arg ref = "bookService"/>
        <constructor-arg ref = "accountsService" />
</bean>
```

Conclusion

- We explored injecting object dependencies in Spring.
- We saw two alternatives for injecting objects:
  - Setter injection
  - Constructor injection
- XML configuration helps define and inject dependencies efficiently.