Spring - Fler koncept inom containerhantering - Övningar

Mål

I denna övning kommer vi att:

- Lära oss hur vi kan injicera hela objekt i Spring.
- Använda konstruktionsbaserad injection.
- Konfigurera beroenden i application.xml.

Steg 1: Skapa ett nytt interface för hantering av Account

Vi fortsätter att arbeta med BookStore-projektet. Förutom *BookService* ska vi nu skapa ett nytt interface *AccountsService*.

- Skapa ett nytt interface *AccountsService* som ska hantera alla ekonomiska transaktioner i systemet.
    - Lägg till en metod *raiseInvoice* som skapar en faktura för en kund *(raiseInvoice(Book requiredBook))*
- Implementera AccountsService i en klass: *AccountsServiceMockImpl.*
    - *public void raiseInvoice(Book requiredBook){*

        System.out.println("Raised the invoice for " + requiredBook); },
- Konfigurera en mock-implementation av tjänsten.

Varför?

- Vi behöver en tjänst som hanterar betalningar och fakturering i systemet.
- Genom att använda en mock-implementation kan vi testa systemet innan en riktig databas kopplas in.

Steg 2: Skapa en tjänst för bokköp

Scenario

Nu ska vi skapa en metod *buyBook* som:

- Använder *BookService* för att hitta den bok som kunden vill köpa.
- Använder *AccountsService* för att skapa en faktura för köpet.

Skapa ett nytt gränssnitt PurchasingService

- Skapa ett nytt interface *PurchasingService.*
- Lägg till en metod *buyBook* som tar emot en ISBN-kod som parameter.
- Implementera PurchasingService i en ny klass (*PurchasingServiceImpl)*
- I metoden buyBook:
    - Hämta boken genom att anropa *BookService.*
    - Skapa en faktura genom att anropa *AccountsService*.

Varför?

- Det blir enklare för klienten att bara anropa en metod för att köpa en bok.
- Vi följer principen att programmera mot interface.

Steg 3: Använda Dependency Injection

I vår implementation av *PurchasingService* ska vi inte hårdkoda implementationerna av *BookService och AccountsService*.

- Skapa två privata instansvariabler för *BookService och AccountsService* i *PurchasingServiceImpl*.
- Skapa setter-metoder för att injicera dessa beroenden.

Steg 4: Konfigurera Spring Container i application.xml

- Lägg till en bean för *AccountsServiceMockImpl.*
- Lägg till en bean för *PurchasingServiceImpl.*
- Injicera beroenden genom att använda <property>-taggar i application.xml.
- Testa att köra koden.

Vad du ska testa:

- Om du inte injicerar beroenden korrekt, kommer du att få en NullPointerException.
- Om koden fungerar korrekt ska du få en utskrift om att en faktura har skapats.

Steg 5: Alternativ metod - Konstruktorbaserad injection

Vi har använt setter-baserad injection, men nu ska vi testa konstruktionsbaserad injection.

- Ta bort setter-metoderna i *PurchasingServiceImp*l.
- Lägg till en konstruktor som tar emot *BookService och AccountsService* som parametrar.
- Uppdatera application.xml genom att använda <constructor-arg> istället för <property>.
- Kör koden och verifiera att den fungerar.