Spring Container Övningar

Mål

I denna övning kommer vi att:

- Skapa en enkel applikation som skriver ut dagens meddelande.
- Arbeta med Spring Containers.
- Skapa en XML-konfigurationsfil för att styra vilka klasser som klienten ska instansiera och anropa.

Repetition

Tidigare har vi pratat om koppling (coupling) och sett att för att minska kopplingen måste vi göra tre saker:

1. Programmera mot Interface.
2. Använda Dependency Injection.
3. Centrera konfigurationen.

Vi har redan gjort steg 1 och 2 i ett tidigare kapitel. Nu ska vi se hur vi kan använda en centraliserad konfiguration genom att använda containers.

Steg 1: Skapa ett nytt projekt

- Ladda ner projektet *MessageProject* och lägg till det i din IDE.
- Lägg till alla bibliotek (JAR-filer) som finns i projektets lib-mapp till class path.

Vad vi ska göra

- Skapa ett interface *MessageOfTheDayService.*
- Implementera interfacet i en klass.
- Konfigurera en Spring Container som skapar och hanterar objekt.
- Hämta och använda objekt från containern.

Steg 2: Skapa ett interface och en implementation

- Skapa ett interface som definierar en metod för att hämta dagens meddelande.
    - *String getTodaysMessage()*
- Implementera interfacet i en klass (*MessageOfTheDayBasicImpl*) som innehåller en privat variabel för meddelandet.
- Skapa en setter-metod för att sätta meddelandet i klassen.
- Implementera metoden *getTodaysMessage* så att den returnerar det sparade meddelandet.

Steg 3: Använd Spring Container

- I stället för att klienten skapar objekt direkt, låt Spring Container skapa objekten.
    - Det finns en fil i src-mappen som heter *application.xml*. Vi lägger till koden mellan <beans> taggarna:
- I XML-filen:
    - Deklarera ett bean för implementationen av *MessageOfTheDayService*.
    - Sätt ett id (*msgService*) för beanen.
        - <bean id="msgService"
            class="*namnet på den implementerande klassen*">
    - Konfigurera egenskapen för meddelandet.
        - <property name="message" value="Hello Spring"/>
- Klienten ska nu hämta objektet från Spring Container istället för att skapa det direkt.
    - Skapa klassen *ClientApplication* som klientkoden.
- Skapa en ClassPathXmlApplicationContext i main metoden i *ClientApplication* som läser in application.xml.
- Hämta objektet från containern med getBean() och anropa metoden för att hämta dagens meddelande.
- Skriv ut meddelandet.

Steg 4: Alternativ implementation

- Skapa en ny implementation av *MessageOfTheDayService* som genererar meddelandet dynamiskt (*MessageOfTheDayDynamicImpl)*
- I stället för att ha ett fast meddelande, skapa en lista med meddelanden för varje veckodag.

  ```
  private String[] messages= new String [] {
                                  "Monday message",
                                  "Tuesday message",
                                  "Wednesday message",
                                  "Thursday message",
                                  "Friday message",
                                  "Saturday message",
                                  "Sunday message",
  };
  ```

- Implementera metoden så att den returnerar meddelandet för dagens veckodag.
  - Denna metod hämtar aktuell veckodag med *GregorianCalendar* och använder den för att indexera messagesList. Det är viktigt att notera att koden använder day - 2 för att anpassa Calendar.DAY_OF_WEEK (som börjar på 1 för söndag) till array-indexeringen (som börjar på 0).
  - Tips:
    - int day = new GregorianCalendar().get(Calendar.DAY_OF_WEEK);
    - String message = messages[day-2];
- Uppdatera *application.xml* för att använda den nya implementationen.
  - Ta bort den fasta property-egenskapen i XML-filen eftersom meddelandena nu genereras i klassen.
- Kör koden och kontrollera att dagens meddelande skrivs ut korrekt.

Steg 5: Injecta värden med Spring

- Istället för att hårdkoda meddelandena i klassen, skapa en tredje implementation (`MessageOfTheDayConfigurableImp`)

- Deklarera en privat array av strängar som lagrar meddelandena för varje dag.
- Lägg till en setter-metod för att ta emot en lista med meddelanden.
- Implementera *getTodaysMessage()* metoden som returnerar dagens meddelande. Den hämtar aktuell veckodag med GregorianCalendar och använder den för att indexera messagesList. Det är viktigt att notera att koden använder *day - 2* för att anpassa Calendar.DAY_OF_WEEK (som börjar på 1 för söndag) till array-indexeringen (som börjar på 0).
    - Tips;
        - int day = new GregorianCalendar().get(Calendar.DAY_OF_WEEK);
        - String message = messages[day-2];
- Uppdatera application.xml för att:
    - Använda den nya implementationen.
    - Skapa en property med en list av meddelanden.
- Kör koden och verifiera att meddelandena injiceras korrekt.

Steg 6: Optimera klientkoden

- Istället för att använda getBean() med ett id, hämta objektet direkt genom att ange typen.
- Kör koden och kontrollera att den fungerar.
- Fundera över problem som kan uppstå om flera implementationer av samma interface finns.