Programowanie komponentowe Spring

Ćw.2 Wiązanie komponentów

Wstęp

Celem zajęć jest utworzenie komponentów Spring z dostarczonych klas Java. Komponenty zostaną następnie skonfigurowane i powiązane aby potem utworzyć dla nich testy jednostkowe.

Konfiguracja komponentów Springa może być zrealizowana na trzy sposoby:

1. Konfiguracja za pomocą adnotacji java
2. Konfiguracja w pliku xml
3. Konfiguracja automatyczna

Wiązanie pomiędzy komponentami jest realizowane przez:

- wstrzykiwanie referencji do konstruktora komponentu,
- wstrzykiwanie referencji do właściwości komponentu.

Część 1 Wiązanie komponentów - rozdział 2 SwA

Na zajęciach utworzymy komponenty Spring z dostarczonych klas javy, odpowiednio skonfigurujemy i powiążemy je w obydwa sposoby a następnie utworzymy dla nich testy jednostkowe.

Konfiguracja komponentów Springa może być zrealizowana na trzy sposoby.

1. Konfiguracja za pomocą adnotacji java
2. Konfiguracja w pliku xml
3. Konfiguracja automatyczna

Wiązanie pomiędzy komponentami jest realizowane przez:

- wstrzykiwanie referencji do konstruktora komponentu,
- wstrzykiwanie referencji do właściwości komponentu.

Zadanie 1 - Konfiguracja Java

Konfiguracja za pomocą adnotacji Java.

- klasa konfiguracyjna powinna być oznaczona adnotacją **@Configuration** i zawierać metody tworzące komponenty oznaczone adnotacją **@Bean**. Te metody pozwalają na wstrzykiwanie komponentów jako parametru konstruktora.
- wciśnięcie kombinacji klawiszy Ctrl + Shift + O uruchamia funkcję automatycznego dodawania niezbędnych importów do klasy java
- utworzenie kontekstu

  AnnotationConfigApplicationContext context =

new AnnotationConfigApplicationContext(
***nazwa_klasy_konfiguracyjnej*.class);**

albo ogólniej

AnnotationConfigApplicationContext context =
new AnnotationConfigApplicationContext();
context.scan("soundsystem");
context.refresh();

druga wersja wyszukuje klasy konfiguracyjne w pakietach o podanych nazwach

- pobrać z kontekstu referencję do komponentu
  komponent = context.getBean(*nazwa_klasy_komponentu.*class);
- na koniec aplikacji kontekst należy zamknąć przez użycie metody **context.close();**
- wiązanie komponentów przez parametr konstruktora jest realizowane w klasie konfiguracyjnej w metodach oznaczonych adnotacją @**Bean** np.
  return new BraveKnight(quest());
- wiązanie komponentów przez właściwości komponentu jest realizowane przez oznaczenie wiązanej właściwości komponentu adnotacją @**Autowired** np.

```
public class BraveKnight implements Knight {
  @Autowired
  private Quest quest;
 ...
```

- klasa testów jednostkowych komponentu Spring wymaga dwu adnotacji:
  @**RunWith(SpringJUnit4ClassRunner.class)** - oznacza, że klasa używa wsparcia Spring dla testów JUnit
  @**ContextConfiguration(classes=*nazwa_klasy_konfiguracyjnej*.class)** - określa klasę konfiguracyjną dla kontekstu Spring

Proszę zaimportować projekt stereo1. Projekt zawiera działającą klasę CDPlayerMain, pustą CDPlayerConfig, działający test jednostkowy CDPlayerTest i jedną płytę cd.
Proszę usunąć ścisłe powiązania między klasami i uzupełnić projekt o:

- konfigurację java z komponentami cd i player oraz powiązanie player z cd
- CDPlayerMain.java - usunąć bezpośrednie wywoływania konstruktorów, utworzyć kontekst, pobrać z niego komponent player, uruchomić odtwarzanie, zamknąć kontekst.
- Zmienić powiązanie między player a cd z wiązania przez konstruktor na wiązanie przez właściwość.
- uzupełnić klasę testów jednostkowych o adnotacje wymagane dla testu jedn. komponentu Spring,
- usunąć metodę testu jednostkowego oznaczoną @**Before** oraz ustawiane przez nią właściwości, zastąpić ją przez:
  @**Autowired**
  private MediaPlayer player;
- przetestować komponent.

Zadanie 2 - Konfiguracja XML

Konfiguracja za pomocą pliku XML.

- szkielet pliku XML znajduje się w pliku src\main\resources\META-INF\spring\stereo.xml
- komponenty są oznaczane przez umieszczenie elementu
  <bean id="*nazwa_komponentu*" class="*nazwa_pakietu.nazwa_klasy*"/>
- powiązanie przez parametr konstruktora - wewnętrzny element
  <constructor-arg ref="*nazwa_komponentu*" />
  np.

```
<bean id="knight" class="sia.knights.BraveKnight">
  <constructor-arg ref="quest" />
</bean>
```

- powiązanie przez właściwość - wewnętrzny element
  <property name="*nazwa_właściwości*" ref="*nazwa_komponentu*"/>
  np.

```
<bean id="knight" class="sia.knights.BraveKnight">
  <property name="quest" ref="quest" />
</bean>
```

- utworzenie kontekstu

```
ClassPathXmlApplicationContext context =
        new ClassPathXmlApplicationContext(
            "META-INF/spring/nazwa_pliku.xml");
```

- pobrać z kontekstu referencję do komponentu przez klasę

```
komponent = context.getBean(nazwa_klasy_komponentu.class);
```

  lub przez nazwę

```
komponent = context.getBean("nazwa_komponentu");
```

- na koniec aplikacji kontekst należy zamknąć przez użycie metody context.close();
- test jednostkowy komponentu Spring wymaga adnotacji:
  @RunWith(SpringJUnit4ClassRunner.class) - oznacza, że klasa używa wsparcia Spring dla testów JUnit
  @ContextConfiguration(locations = "classpath:META-INF/spring/*nazwa_pliku*.xml") - określa plik konfiguracyjny XML dla kontekstu Spring

Proszę zaimportować projekt stereo2, usunąć ścisłe powiązania między klasami
i uzupełnić projekt o:

- konfigurację XML z komponentami cd i player oraz powiązanie player z cd
- CDPlayerMain.java - usunąć bezpośrednie wywoływania konstruktorów, utworzyć kontekst, pobrać z niego komponent player, uruchomić odtwarzanie, zamknąć kontekst.
- Zmienić powiązanie między player a cd z wiązania przez konstruktor na wiązanie przez właściwość. Uwaga, wymaga to utworzenia konstruktora bezparametrowego oraz getera i

setera dla właściwości cd klasy Player. Dzięki temu uzyskujemy zgodność z wymaganiami dla komponentu Java Beans.

- usunąć metodę @Before testu jednostkowego, uzupełnić go o adnotacje wymagane dla testu jedn. komponentu Spring w pliku XML, przetestować komponent.

Zadanie 3 - Konfiguracja automatyczna

Automatyczne wykrywanie i wiązanie komponentów.

- Konfiguracja automatyczna polega na włączeniu automatycznego wykrywania komponentów, przez użycie adnotacji @ComponentScan w klasie zawierającej konfigurację java lub umieszczenie elementu
  <context:component-scan base-package="*nazwa_pakietu_do_przeszukania*" />
  w konfiguracji xml.
- Komponenty - klasy java do wyszukania powinny być oznaczone adnotacją @**Component**
- automatyczne wiązanie komponentów jest realizowane przez oznaczenie adnotacją @**Autowired** wiązanej właściwości lub konstruktora z parametrem komponentu

Proszę zaimportować projekt stereo3 i uzupełnić projekt o:

- konfigurację z automatycznym wykrywaniem i wiązaniem komponentów przez konstruktor
- CDPlayerMain.java - utworzyć kontekst, pobrać z niego komponent player, uruchomić odtwarzanie, zamknąć kontekst.
- Zmienić powiązanie między player a cd z wiązania przez konstruktor na wiązanie przez właściwość.
- Uruchomić test jednostkowy komponentu.

Zadanie 4 - Konfiguracja mieszana

- importowanie konfiguracji java w konfiguracji java
  @**Import(*nazwa_klasy.class*)**
- importowanie pliku konfiguracyjnego xml w konfiguracji java
  @**ImportResource("classpath:*nazwa_pliku.xml*")**
- importowanie konfiguracji java w pliku konfiguracyjnym xml
  <bean class="*nazwa_pakietu.nazwa_klasy*"/>
- importowanie pliku konfiguracyjnego xml w pliku konfiguracyjnym xml
  <import resource="*nazwa_pliku.xml*"/>

Proszę zaimportować projekt stereo4 i uzupełnić projekt o:

- utworzyć osobne konfiguracje dla cd i playera i połączyć je importując - najlepiej do głównej konfiguracji (np. główna konf. java i dwie importowane xml)
- uruchomić testy jednostkowe

Część 2 Zaawansowane wiązanie komponentów - rozdział 3 SwA

Zadanie 5 - Kwalifikatory i profile

- Wstrzykiwanie literałów (wartości) przez konstruktor
  wywołanie konstruktora z odpowiednimi wartościami
- Wstrzykiwanie wartości zewnętrznych
  - utworzenie pliku properties
    nazwa.parametru="wartość"
  - adnotacja w klasie konfiguracyjnej
    @PropertySource("*classpath:/nazwa_pakietu/nazwa_pliku.properties*")
  - wstrzyknięcie komponentu środowiska
    @Autowired
    Environment env;
  - pobieranie wartości z pliku properties
    env.getProperty("*nazwa_parametru*")
- Rozwiązanie niejednoznaczności automatycznych powiązań
  - kwalifikator
    @Primary
  - własne kwalifikatory np. kwalifikator **@Pop**
    @Target({ElementType.CONSTRUCTOR, ElementType.FIELD,
    ElementType.METHOD, ElementType.TYPE})
    @Retention(RetentionPolicy.RUNTIME)
    @Qualifier
    public @interface Pop { }
- Oznaczenie, w którym profilu komponent ma być utworzony
  @Profile("*nazwa_profilu*")
- Ustawienie aktywnego profilu (jedna z kilku możliwości)
  System.setProperty("spring.profiles.active", "*nazwa_profilu*");

Proszę zaimportować projekt stereo5 i:

- utworzyć 3 różne komponenty cd wstrzykując nazwę płyty i wykonawcy do konstruktora klasy BlankDisc przez java config, w tym jeden z pliku properties
- rozwiązać niejednoznaczość powiązania odtwarzacza z płytą
  - użyć do tego adnotacji @Primary
  - użyć do tego własnego kwalifikatora np. @Pop do oznaczenia komponentu i oznaczenia właściwości wstrzykiwanej przez kontener
  - użyć do tego kwalifikatorów np. @Qualifier("AvantGarde") do oznaczenia komponentu i oznaczenia właściwości wstrzykiwanej przez kontener
  - użyć do tego nazw profili: dev, test, prod
- dla każdego z profili wiązać inną płytę cd z playerem.
- sprawdzić działanie przełączając profile oraz spróbować napisać testy jednostkowe

Zadanie 6 - Wyrażenia SpEL

Proszę samemu wymyślić, jak można użyć Spring Expression Language w projekcie stereo6. Mogą być zarówno najprostsze, jak i bardziej złożone zastosowania.