Zaawansowane Aplikacje Internetowe

Spring framework

Piotr Mazur

Katedra Mikroelektroniki i Technik Informatycznych

Łódź, 26 października 2010

1 Spring Framework

Spring Framework

Framework dostarczający między innymi:

- Kontener IoC (Inversion of Control)
- Obsługę warstwy dostępu do danych (Hibernate, JPA ...)
- Deklaratywne zarządzanie transakcjami
- Obsługę wywołań zdalnych (Hessian/Burlap, RMI, SOAP)
- Harmonogram zadań (Quartz)
- Obsługę przesyłania wiadomości (Email, JMS)
- Obsługę modelu MVC

Zarządzanie obiektami

Spring Framework dostarcza kontener zarządzający obiektami aplikacji. Definiowanie nowych beanów w aplikacji może odbywać się poprzez konfigurację (plik XML) lub skanowanie połączone z annotacjami

- Każdy obiekt zarządzany przez Spring Framework może posiadać pre oraz postprocesory umożliwiające między innymi inicjalizację tych obiektów
- Dodatkowo obsługiwane jest także niszczenie obiektów (mechanizm podobny do działania destruktorów w innych językach programowania)
- Każdy obiekt musi posiadać nazwę **(Bean Name)**
- Obiekty przechowywane są w tzw: Kontekście Aplikacji **(Application Context)**
- Istnieje kilka implementacji kontekstu aplikacji różniących się głównie sposobem konfiguracji

ApplicationContext

Utworzenie kontekstu aplikacji odczytującego plik konfiguracyjny znajdujący się w classpath

Przykład utworzenia kontekstu aplikacji

```
ApplicationContext context=new ClassPathXmlApplicationContext("applicationContext.xml");
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:aop="http://www.springframework.org/schema/aop"
   xmlns:tx="http://www.springframework.org/schema/tx"
   xsi:schemaLocation="
   http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2
   http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.0.xsd
   http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.0.xsd"

   <bean id="myBean" class="org.app.Test" />
</beans>
```

ApplicationContext

Pobranie zkonfigurowanego beana z kontekstu aplikacji

```
Test bean=(Test)context.getBean("myBean");
```

Po pobraniu beana można z niego korzystać tak jak z każdego innego obiektu.
Zaletą takiego rozwiązania jest przechowywanie wszystkich obiektów aplikacji w jednym, centralnym repozytorium (kontekście aplikacji)

Inversion of Control

Mechanizm **Inversion of Control** umożliwia obsługę wstrzykiwania zależności **(Dependency Injection)**

- Obiekty zależne są automatycznie **Wstrzykiwane** do klas, o ile obiekt danego typu istnieje w kontekście aplikacji
- Pozwala to na przesunięcie logiki związanej z wypełnianiem obiektów do konfiguracji aplikacji
- Przykład: Wstrzykiwanie innej implementacji klasy serwisowej lub klasy dostępu do danych **(DAO)**

Inversion of Control

Przykład wstrzykiwania zależności

```
<bean id="testDAO" class="Test.TestDAO" />
<bean id="testService" class="Test.TestService" >
   <property name="testDao" ref="testDAO"></property>
</bean>
```

Jeżeli klasa **Test.TestService** posiada pole o nazwie **testDAO** zostanie ono automatycznie wypełnione obiektem klasy **Test.TestDAO**

Inversion of Control

Beany można konfigurować także za pomocą automatycznego skanowania pakietów oraz annotacji

```
<context:component-scan base-package="Test" />
```

Należy oczywiście uwzględnić przestrzeń nazw **context** w pliku konfiguracyjnym

```
xmlns:context="http://www.springframework.org/schema/context"
```

Inversion of Control

Jeżeli korzystamy ze skanowania, definiowanie beanów może odbywać się za pomocą annotacji

- **@Component** - Definiowanie beana
- **@Autowired** - Automatyczne wypełnianie pola jeżeli istnieje bean takiej samej klasy
- **@Scope** - Definiowanie zakresu beana (Singleton, prototype)
- **@PostConstruct** - uruchomienie metody inicjalizacyjnej. Metoda zostanie uruchomiona po wypełnieniu wszystkich pól

Integracja z serwerem aplikacyjnym

Dodanie klasy automatycznie inicjalizującej kontekst aplikacji.

web.xml

```
<listener>
   <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

Kontekst aplikacji może zostać automatycznie wstrzyknięty do dowolnego beana poprzez implementację interfejsu **ApplicationContextAware**

Integracja ze Struts 2

Plugin **Spring** pozwala na automatyczne rejestrowanie akcji jako beanów frameworku **Spring**.
Każda akcja podlega wtedy takim samym działaniom jak typowy bean (np: działają annotacje **@Autowired** etc..)