infoShare ACADEMY

Invest in Pomerania ACADEMY

INVEST IN POMERANIA ACADEMY

Fundusze Europejskie Program Regionalny

Rzeczpospolita Polska

URZĄD MARSZAŁKOWSKI WOJEWÓDZTWA POMORSKIEGO

Unia Europejska Europejski Fundusz Rozwoju Regionalnego

infoShare ACADEMY

Invest in Pomerania ACADEMY

Aplikacja Webowa: Spring Core

Fundusze Europejskie Program Regionalny
Rzeczpospolita Polska
URZĄD MARSZAŁKOWSKI WOJEWÓDZTWA POMORSKIEGO
Unia Europejska Europejski Fundusz Rozwoju Regionalnego

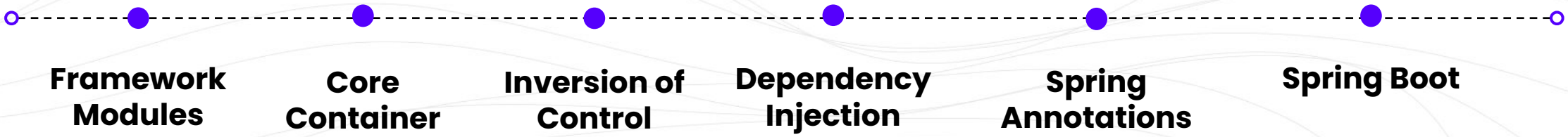
HELLO

Wojciech Rozwałka

Java Full-Stack Developer

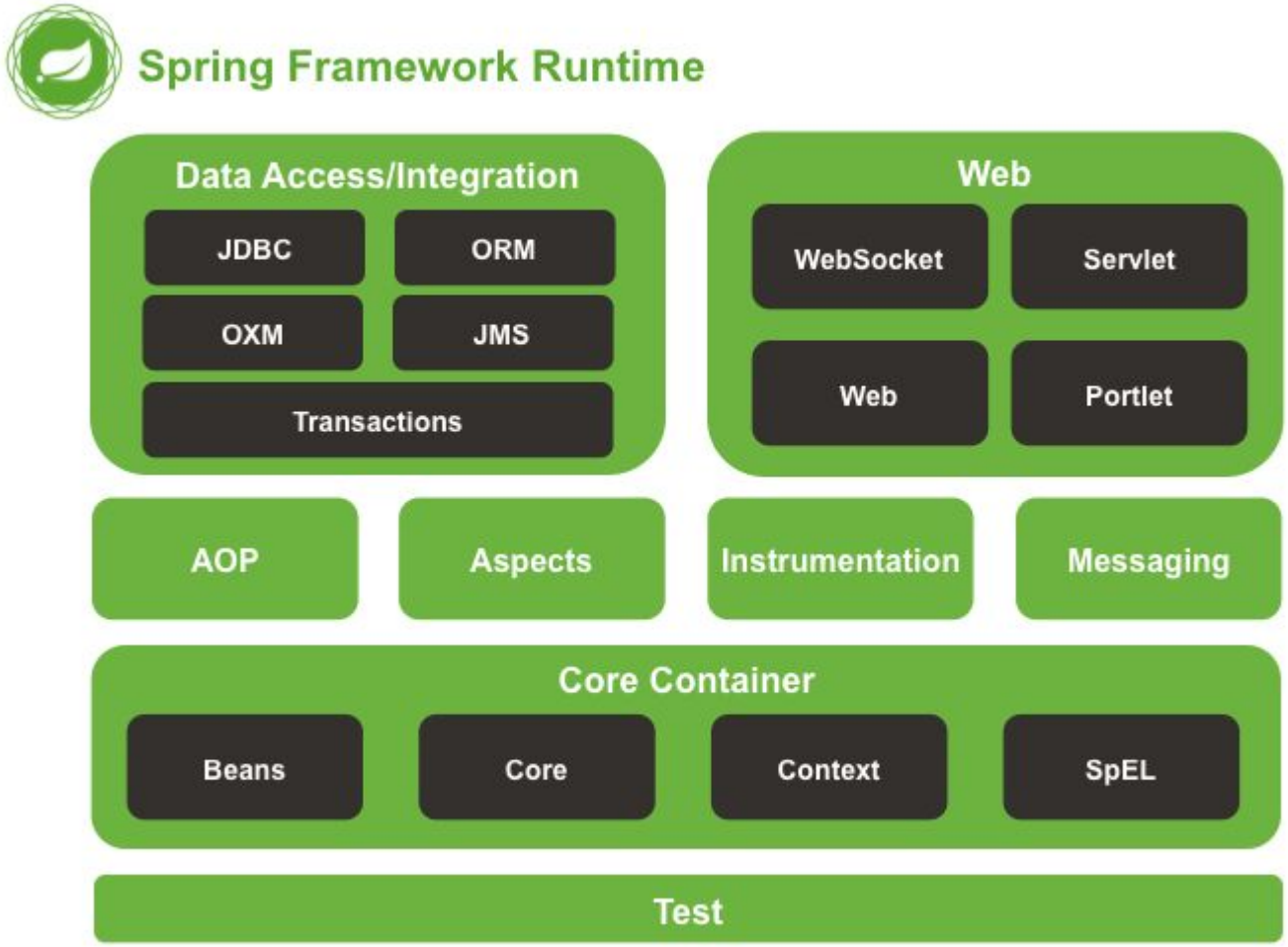
InfoShare ACADEMY

Agenda



Framework Modules

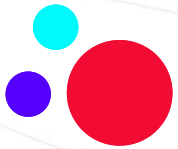
Spring Framework zawiera około 20 modułów m.in. Core Container, AOP, Data Access, Web etc.



Core Container

Zawiera w sobie jedne z najistotniejszych modułów:

- spring-core
- spring-beans
- spring-context
- spring-expression



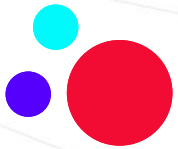
Core Container

Core i Bean - podstawowa część Springa, zawiera funkcjonalności kontenera zależności “Inversion of Control” (IoC) oraz “Dependency Injection (DI), ale więcej o tym później :)
Pozwala na oddzielenie inicjalizacji i konfiguracji zależności od logiki biznesowej.
Jest to implementacja wzorca fabryki.

Context - rozbudowuje funkcjonalności dostarczone przez moduły Core i Bean. Dodaje m. in. wsparcie internalizacji, dostęp do zasobów URL i plików, propagację eventów do beanów, ładowanie hierarchicznych kontekstów, EJB, JMX.

Expression Language - język wyrażeń pozwalający na dostęp do informacji w beanach. Definiuje tzw. “unified expression language” określony w specyfikacji JSP 2.1.

https://docs.spring.io/spring-framework/docs/4.3.12.RELEASE/spring-framework-reference/html/overview.html

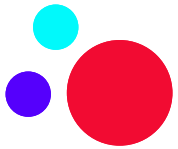


Inversion of Control

Kontener IoC pozwala na przechowywanie obiektów i zarządzanie nimi w całym cyklu życia aplikacji.

- nie musisz ręcznie tworzyć obiektów, kontener zrobi to za Ciebie,
- umożliwia wstrzykiwanie zależności – czyli dostarczanie obiektu dokładnie tam gdzie tego potrzebujesz.

To co musimy dostarczyć to jedynie konfigurację, czyli informację na temat w jaki sposób Spring ma tworzyć beany.



Konfiguracja

Adnotacje

Oznaczenie klas adnotacjami, dzięki którym Spring wie jak dany bean ma zostać zakwalifikowany.

Klasa konfiguracyjna

Miejsce w którym dostarczamy definicję beanów za pomocą kodu.

Plik XML

Ustrukturyzowany plik, który zawiera definicje beanów ujętą w znacznikach XML

Konfiguracja

```
package com.infoshareacademy.module.config;
```

Klasa konfiguracyjna

```
@Configuration
public class ApplicationConfig {

    @Bean
    public SpringBean1 springBean1() {
        return new SpringBean1();
    }

    @Bean
    public SpringBean2 springBean2() {
        return new SpringBean2();
    }

}
```

Plik 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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="springBean1" class="com.spring.beans.SpringBean1"/>

</beans>
```

Adnotacje

```
@Bean
public SpringBean1 getSpringBean1() {
    return new SpringBean1();
}
```

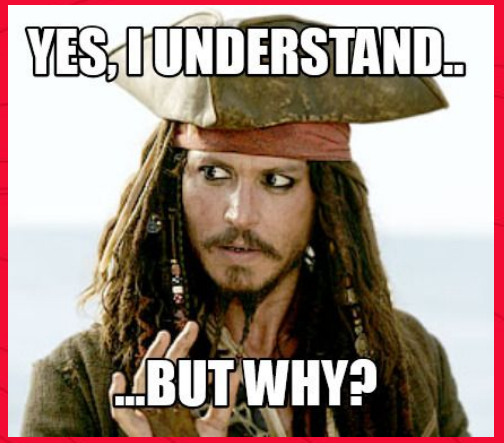
Dependency Injection

Wstrzykiwanie beanów jest banalnie proste :)
Wykorzystujemy do tego adnotację
`@Autowired`
Możemy jej użyć w trzech miejscach:

- na polu w klasie
- metodzie (setter)
- konstruktorze

ZALECANE

Najbardziej popularnym i wskazanym podejściem jest wstrzykiwanie przez konstruktor.



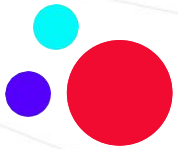
- ułatwia testowanie
- umożliwia zadeklarowanie pół jako final
- zabezpiecza przed NullPointerException

Wstrzykiwanie przez pole

```java
@Component
public class EspressoMaker implements CoffeeMaker {

    @Override
    public void getCoffee() {
        System.out.println("EspressoMaker getCoffee");
    }

}
```

```java
@Component
public class CoffeeVendingMachine {

    @Autowired
    private CoffeeMaker coffeeMaker;

    public void getCoffee() {
        coffeeMaker.getCoffee();
    }

}
```

Live coding



Wstrzykiwanie przez setter

```java
@Component
public class CoffeeVendingMachine {

    private CoffeeMaker coffeeMaker;

    public void getCoffee() {
        coffeeMaker.getCoffee();
    }

    @Autowired
    public void setCoffeeMaker(CoffeeMaker coffeeMaker) {
        this.coffeeMaker = coffeeMaker;
    }

}
```

Zadanie

- Zamień wstrzykiwanie przez pole na wstrzykiwanie przez setter
- Zastanów się czy wstrzykiwanie interfejsu zamiast klasy jest poprawne

EXTRA

- Stwórz drugą implementację interfejsu `CoffeMaker` i sprawdź co się wydarzy

Wstrzykiwanie przez konstruktor

```java
@Component
public class CoffeeVendingMachine {

    private final CoffeeMaker coffeeMaker;

    public CoffeeVendingMachine(CoffeeMaker coffeeMaker) {
        this.coffeeMaker = coffeeMaker;
    }

    public void getCoffee() {
        coffeeMaker.getCoffee();
    }

}
```

Zadanie

- Zamień wstrzykiwanie na wstrzykiwanie przez konstruktor

Adnotacja `@Autowired` istnieje.
Natomiast warto wspomnieć, że od Springa 4.3, adnotacja przy konstruktorze nie jest wymagana. Jest tak tylko i wyłącznie wtedy gdy klasa ma jeden konstruktor.

@Bean

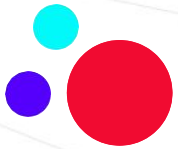
Adnotacja służąca do stworzenia beanów z klasy w przypadku, gdy nie mamy dostępu do kodu źródłowego, lub, gdy chcemy stworzyć kilka beanów tej samej klasy, posiadających różny stan.

```
@Configuration
public class Config {

    @Bean
    public ObjectMapper getObjectMapper() {
        return new ObjectMapper();
    }

}
```

@Primary

W przypadku, gdy jest kilku kandydatów do wstrzyknięcia, adnotacja ta definiuje, który ma pierwszeństwo (domyślna implementacja)

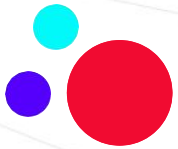
```java
@Component
@Primary
public class EspressoMaker implements CoffeeMaker {

    @Override
    public void getCoffee() {
        System.out.println("EspressoMaker getCoffee");
    }

}
```



@Qualifier

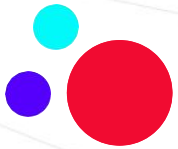
W przypadku kilku kandydatów do wstrzyknięcia, adnotacja ta pozwala określić którą implementację wstrzyknąć.

```java
@Component
public class CoffeeVendingMachine {

    private final CoffeeMaker coffeeMaker;

    public CoffeeVendingMachine(@Qualifier("macchiatoMaker") CoffeeMaker coffeeMaker) {
        this.coffeeMaker = coffeeMaker;
    }

    public void getCoffee() { coffeeMaker.getCoffee(); }

}
```



Zadanie

- Stwórz klasę `MacchiatoMaker` implementującą interfejs `CoffeeMaker`
- Oznacz ją adnotacją `@Qualifier("macchiatoMaker")`
- Dodaj w konstruktorze klasy `CoffeeVendingMachine` w.w. adnotację



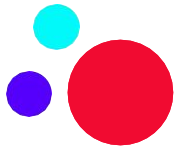
Adnotacje tworzące beany

- `@Component` – tworzy bean zarządzany przez kontener Springa
- `@Repository` – tworzy bean zarządzany przez kontener Springa w warstwie dostępu do danych (metody obsługujące bazę danych)
- `@Service` – tworzy bean zarządzany przez kontener Springa w warstwie serwisowej (powinien zawierać logikę biznesową)
- `@Controller` – tworzy bean zarządzany przez kontener Springa w warstwie webowej

Technicznie rzecz biorąc `@Repository, @Service, @Controller` nie różnią się od `@Component`. Pozwala to, zarówno nam jak i Springowi, określić odpowiedzialność beana.

Zadanie

- Stwórz klasę `CoffeeController`
- Stwórz klasę `CoffeeServiceImpl` implementującą interfejs `CoffeeService`
- Stwórz klasę `CoffeeRepository`
- Wstrzyknij odpowiednie klasy wg. modelu MVC oznaczając je odpowiednimi adnotacjami



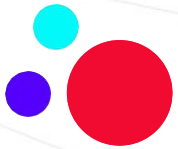
Spring Boot

infoShareAcademy.com

infoShare ACADEMY

Spring Boot

Spring Boot jest narzędziem do tworzenia aplikacji, opartym na poszczególnych modułach framework'a Spring. Spring Boot dostarcza domyślne, najbardziej użyteczne zdaniem społeczności konfiguracje modułów Spring. Jest oparty o zasadę convention over configuration.



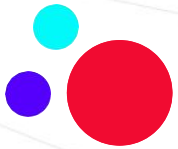
Convention over configuration

Rozwiązanie pozwalające tworzyć zaawansowane aplikacje przy stosunkowo niewielkiej ilości kodu logiki aplikacji oraz mechanizmów konfiguracji. Konfiguracja wykorzystuje najpopularniejsze przypadki wykorzystania, co pozwala uruchomić rozwiązanie w kilka minut. Nadpisuje się jedynie domyślne ustawienia i rozszerza.

Spring Initializr

Narzędzie pozwalające wygenerować pusty projekt oparty o Springa: Spring Initializr

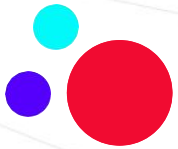
Możliwe też wygenerowanie projektu w IntelliJ (wersja Ultimate).



@SpringBootApplication

Adnotacja ułatwiająca konfigurację aplikacji. Jest to zbiorcza adnotacja, która zawiera w sobie:

- `@SpringBootConfiguration`
- `@ComponentScan`
- `@EnableAutoConfiguration`



EXTRA Bean Cycle

```
package com.infoshareacademy.module.cycle;
```

EXTRA Bean Scopes

4.4 Bean scopes

```
package com.infoshareacademy.module.scopes;
```

Dzięki za uwagę

Fundusze Europejskie Program Regionalny

Rzeczpospolita Polska

URZĄD MARSZAŁKOWSKI WOJEWÓDZTWA POMORSKIEGO

Unia Europejska Europejski Fundusz Rozwoju Regionalnego

Dodatkowe materiały

- Spring - Dokumentacja
- Spring Boot - Dokumentacja
- Baeldung
- Spring Guru