



Spring Boot 101

Wprowadzenie

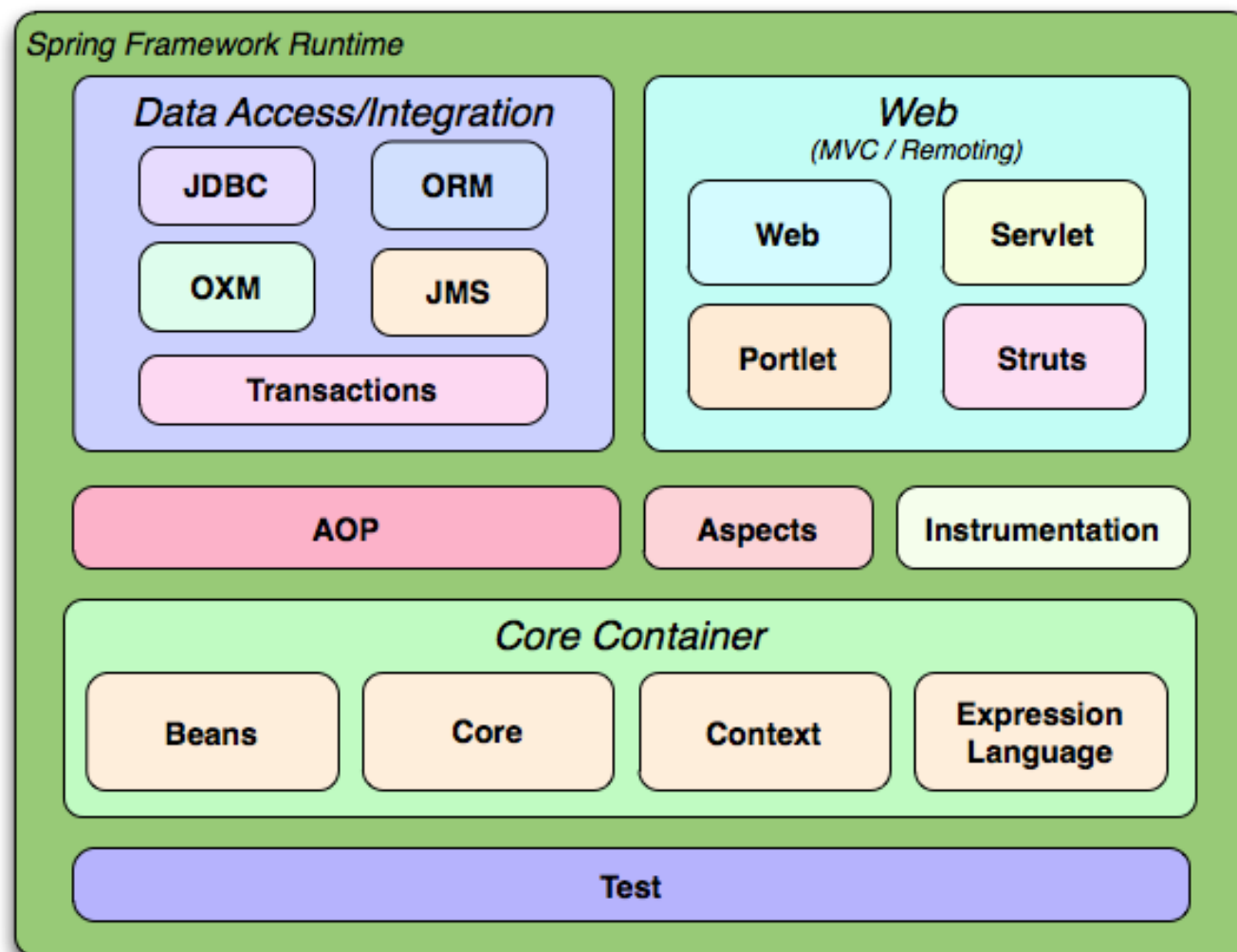
Zaawansowane Zagadnienia Programowanie w Javie – Edycja 2023

- **Jakarta** to rozszerzenie standardowej Javy skierowane do rozwiązań komercyjnych, które dostarcza:
 - Obsługę transakcyjności operacji (JTA)
 - Zapewnienie bezpieczeństwa
 - Obsługę komunikacji za pomocą komunikatów (JMS)
 - Dostępu do danych za pomocą mapowania relacyjnych obiektów (JPA)
 - Definiowanie UI (Servlets, JSP, JSF)
 - Łączenie komponentów, które implementują logikę biznesową (EJB)
- Aplikacje są uruchamiane na serwerze aplikacyjnym (np. popularny *Tomcat*, *JBoss*, *Glassfish*)
- Głównym argumentem krytycznym jest tworzenie przez programistę powtarzających i podatnych na błędy elementów kodu (*boilerplate code*), konfigurowanie i administrowanie serwerów, brak stabilności i zrozumiałej dokumentacji
- Od 2017 roku firma *Oracle* przekazała wsparcie i rozwój Javy EE do *Eclipse Foundation* bez prawa do nazwy, stąd nastąpiło przemianowanie na Jakarta EE





- Szkielet tworzenia aplikacji – framework na platformę Java
- Dostarcza rozwiązania dzięki którym możemy się skupić na rozwoju naszej aplikacji



Co jeszcze potrafi Spring?



Wspracie dla
mikroserwisów



Serverless



Aplikacje webowe



Batch



Cloud



Reactive



Event Driven



Spring Framework

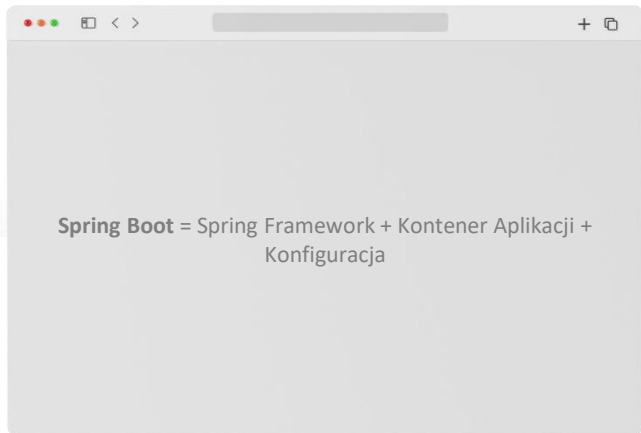
Narzędzie znacznie skracające proces tworzenia aplikacji

- Dostarcza sprawne mechanizmy przy zachowaniu najlepszych praktyk programistycznych
- Zachowuje określone standardy uporządkowujące kod

Spring Boot

Framework dla frameworka

- Dostarcza domyślnie skonfigurowaną aplikację Spring
- Wykorzystuje najlepsze praktyki Springa



Spring Boot = Spring Framework + Kontener Aplikacji + Konfiguracja

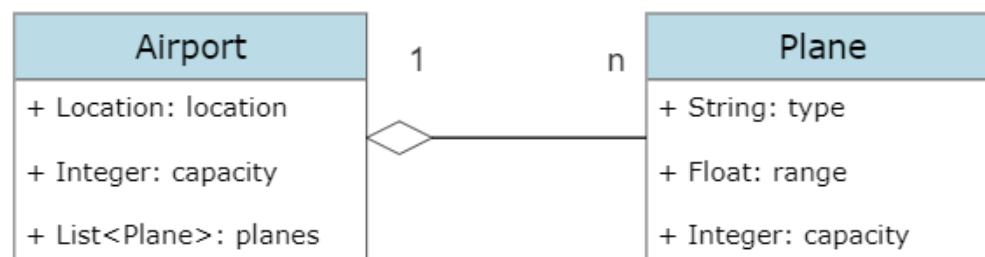
- » Inversion of Control
- » Dependency Injection
- » IoC Container
- » Convention over Configuration

- ***Odwrócenie Kontroli*** to reguła **mówiąca o przeniesieniu odpowiedzialności** wywołania metod oraz tworzenia obiektów na **szkielet aplikacyjny** (framework).
- Jedną z form *Odwrócenia Kontroli* jest ***Wstrzykiwanie Zależności***.

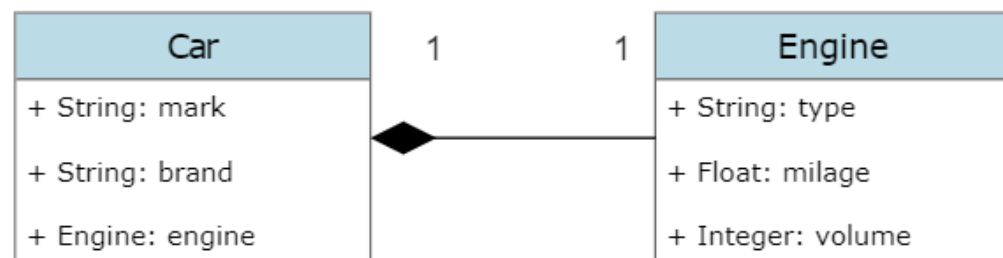
- **Wstrzykiwanie Zależności** jest wzorcem projektowym, którego celem jest **zapewnienie luźnego połączenia między obiektami** (usunięcie bezpośrednich zależności między nimi).
- Jest to możliwe poprzez wstrzyknięcie **instancji utworzonych obiektów do innego obiektu**, który korzysta z ich właściwości.



Agregacja – luźne powiązanie obiektów



Kompozycja - ścisłe powiązanie obiektów



Przekazywanie obiektu **A** jako:

- Parametr **konstruktora**,
- Z wykorzystaniem metody **set**,
- Bezpośrednie ustawienie wartości

do obiektu **B**, który posiada pole **A**.

```
Car car = new Car("Audi", "A4", 150000L);
```

```
//Dependency injection by Constructor
```

```
Garage garage = new Garage(car);
```

```
//Dependency injection by Setter
```

```
garage.setCar(car);
```

```
//Dependency injection by Value
```


```
garage.car = car;
```

A gdzie w tym wszystkim Spring?

Spring zapewnia mechanizm umożliwiający wstrzyknięcia zależności w sposób „*auto-magiczny*” z wykorzystaniem adnotacji



Rdzeń Spring'a. Odpowiada za tworzenie, wiązanie, konfiguracje i zarządzanie cyklem życia obiektów do niego zarejestrowanych



```
@Configuration
public class Config {

    @Bean
    Garage provideGarage() {
        return new Garage();
    }
}
```



@Component – podstawowy stereotyp uwidaczniający za adnotowaną nim klasę dla mechanizmu auto-detekcji komponentów Spring’a, w celu dodania go do kontenera IoC



@Service – specjalizacja adnotacji @Component, ma charakter informacyjny – komponent odpowiedzialny za logikę biznesową



@Controller – specjalizacja adnotacji @Component, ma charakter informacyjny – komponent odpowiedzialny za obsługę zapytań HTTP



@Repository – specjalizacja adnotacji @Component, ma charakter informacyjny – komponent odpowiedzialny za komunikację z warstwą danych oraz przechwytywanie i opakowanie wyjątków



@Bean – adnotacja nadawana nad metodami – implikuje przekazywanie kontroli nad stworzonym w metodzie obiektem kontekstowi (kontenerowi) Spring’a. Domyślnie tworzy Singleton



@Configuration – oznacza komponent (klasę) przechowującą Bean’y



@Autowired – stosowana nad polem, setterem lub konstruktorem w celu umożliwienia Springowi wstrzyknięcie Beana przechwytywanego w kontenerze



@ComponentScan – uruchamia „poszukiwanie” przez Spring’a klas i metod oznaczonych adnotacjami @Bean, @Component i pochodnymi w pakiecie bieżącym oraz w głąb



Zasięg beanów springowych

- ✓ Singleton (domyślny)
- ✓ Prototype
- ✓ Request
- ✓ Session
- ✓ Global session

Rozkład jazdy

1

Zapoznanie ze stroną start.spring.io

2

Prosta aplikacja Spring Boot

3

Rozbudowa aplikacji o komunikację z bazą danych (JPA)

4

Warstwa prezentacji z wykorzystaniem RESTowych webserwisów

5

Actuators – meta informacje

6

Bezpieczeństwo

7

OpenAPI/RestAssured/Thymeleaf



8

Zadania do wykonania