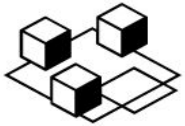
SPRING BOOT

Bartłomiej Leśnicki i Piotr Libucha

Agenda:

1. Czym jest Spring i Spring Boot.
2. Kontener IoC - beany
3. Dependency Injection
4. Najważniejsze moduły Spring Boot (krótki opis kilku)
5. REST - demo GET i POST, fetch z frontu
6. Podstawowe adnotacje @
7. Integracja z innymi technologiami

Spring vs Spring Boot



Microservices

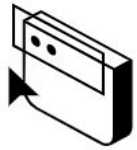
Quickly deliver production-grade features with independently evolvable microservices.

Reactive

Spring's asynchronous, nonblocking architecture means you can get more from your computing resources.

Cloud

Your code, any cloud—we've got you covered. Connect and scale your services, whatever your platform.

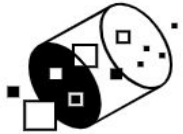


Web apps

Frameworks for fast, secure, and responsive web applications connected to any data store.

Serverless

The ultimate flexibility. Scale up on demand and scale to zero when there's no demand.



Event Driven

Integrate with your enterprise. React to business events. Act on your streaming data in realtime.

Batch

Automated tasks. Offline processing of data at a time to suit you.

Główne różnice między Spring a Spring Boot

1. Kompleksowość vs Prostota
2. Konfiguracja
3. Wbudowany serwer (Spring Boot)
4. Budowa mikroserwisów (Spring Boot)

Kontener IoC

Inversion of Control (IoC) - Odwrócone Sterowanie - przekazanie sterowania nad programem do używanego frameworku.

Głównym kontenerem IoC w Springu jest ApplicationContext.

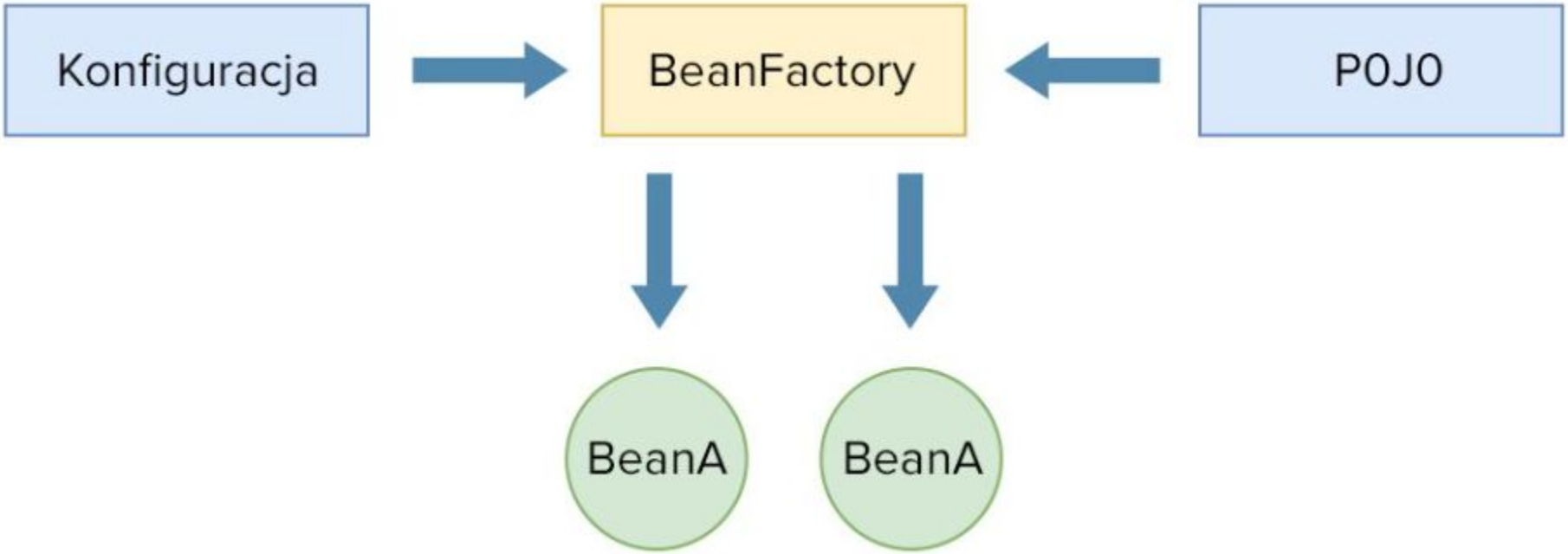
Często stosuje się też lżejszy, mniej funkcjonalny: BeanFactory.

Dla aplikacji webowych używany jest najczęściej WebApplicationContext wspierający servelety.

Spring Boot wspiera również kontenery IoC spoza Sping'a: Google Guice, PicoContainer, Apache DeltaSpike.

Standardowo i w większości przypadków w Spring Boot używany kontener Springa ApplicationContext.

Struktura Kontenera IoC



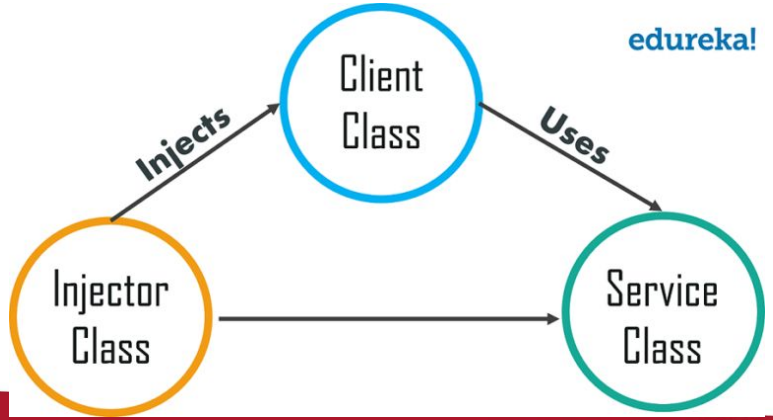
Konfiguracja

W pierwszej kolejności musimy dostarczyć konfiguracje – informacji na temat jak mają być tworzone beany, oraz jak mają się one zachowywać.
Konfiguracje możemy dostarczyć na 3 sposoby:

- Adnotacje
- Klasa konfiguracyjna
- Plik XML (archaiczne, dawno używane)

Dependency injection

Dependency injection (wstrzykiwanie zależności) jest wzorcem projektowym używanym w programowaniu obiektowym w celu osiągnięcia luźniejszego wiązania pomiędzy komponentami aplikacji. Odpowiedzialność za tworzenie obiektów i zarządzanie zależnościami między nimi jest wyodrębniona do innego komponentu zwanego dependency injectorem.



Dependency injection

Wyróżniamy trzy główne sposoby wstrzykiwania zależności:

- Constructor injection
- Setter injection
- Interface injection

W Spring Boot’cie używa się najczęściej constructor injection.

```
@Service
public class MyService {
    private final MyRepository repository;

    @Autowired
    public MyService(MyRepository repository) {
        this.repository = repository;
    }
```

Moduły Spring Boot

- Spring Boot Starter
- Spring Boot Starter Web
- Spring Boot Starter Data JPA
- Spring Boot Starter Security
- Spring Boot Starter Test
- Spring Boot Actuator
- WebSocket
- JDBC driver
- Spring Reactive Web (WebFlux i Netty)

Wiele, wiele innych…

https://start.spring.io/

Integracja z innymi technologiami

Spring Boot umożliwia łatwą integrację z wieloma innymi technologiami:

- Bazy danych
- Technologie frontendowe
- Integracja z usługami chmurowymi
- Mikroserwisy
- RESTful API
- Cache
- Integracja z innymi frameworkami i bibliotekami

Adnotacje

Adnotacje (annotations) są rodzajem metadanych, które dostarczają informacji o klasach, metodach polach, czy argumentach. W Javie adnotacje rozpoczynają się od symbolu @, po którym następuje nazwa adnotacji. Adnotacje mogą mieć wartości i parametry.

Adnotacje w Spring Boot’cie

Podstawowe i najczęściej używane adnotacje w Spring Boot’cie:

- **@SpringBootApplication** - w połączeniu z kilkoma innymi adnotacjami służy do oznaczania głównej klasy aplikacji.
- **@Controller** - oznacza klasę jako kontroler obsługujący żądania HTTP.
- **@ResponseBody** - mówi, że wartość zwracana przez metodę powinna być wysłana do klienta jako odpowiedź na zapytanie.
- **@RestController** - połączenie **@Controller** i **@ResponseBody**.
- **@RequestMapping** - mapuje zapytania HTTP do metod w kontrolerze.

```
@RequestMapping(value = "/hello", method = RequestMethod.GET)
public String helloWorld() {
    return "Hello, World!";
}
```

Adnotacje w Spring Boot’cie

- **`@GetMapping`** - **`@RequestMapping`** wyspecjalizowana dla HTTP GET.

```
@GetMapping("/hello")
public String helloWorld() {
    return "Hello, World!";
}
```

- **`@PostMapping`** - **`@RequestMapping`** wyspecjalizowana dla HTTP POST.

```
@PostMapping("/submit")
public String submitForm(@ModelAttribute("form") MyForm form, Model model) {
    // Save data to /dev/null
    model.addAttribute("message", "Form submitted successfully");
    return "result";
}
```

Adnotacje w Spring Boot’cie

- **`@Bean`** - pozwala na zdefiniowanie beana, który powinien być zarządzany przez Spring. Zazwyczaj używa się w połączeniu z **`@Configuration.`**
- **`@Configuration`** - oznacza klasę konfiguracyjną.

```java
@Configuration
public class MyConfig {

    @Bean
    public MyService myService() {
        return new MyService(myRepository());
    }

    @Bean
    public MyRepository myRepository() {
        return new MyRepositoryImpl();
    }
}
```

Adnotacje w Spring Boot’cie

- **`@Value`** - wstrzykuje wartość z pliku konfiguracyjnego.
- **`@Service`**, **`@Repository`**, **`@Component`** - służą do oznaczania beanów zarządzanych przez Spring, takich jak usługi, repozytoria itp.
- **`@Autowired`** - automatycznie wiąże beany ze sobą.

```java
@Service
public class MyService {

    private final MyRepository repository;

    @Autowired
    public MyService(MyRepository repository) {
        this.repository = repository;
    }
```

Źródła

1.https://azpanel.azilen.com/uploads/everything_must_know_spring_boot_application_scratch_12_8c4e62d4fe.jpg

2.https://www.inovex.de/wp-content/uploads/2021/04/training-spring-boot.png

3.https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/beans.html

4.https://bykowski.pl/spring-context-jak-dziala-kontener-ioc/

5. https://www.edureka.co/blog/what-is-dependency-injection/

6.https://spring.io/