EJB 2.0 -> Spring->Spring Boot czyli najlżejsza wersja

EJB 3.0 przewyższa Springa ale na szczęście powstał Spring Boot (a więc zarządzanie przez pluginymavenowe)

**Spring Boot to**

To podział na paczki i zarządzanie by nie dochodziło do konfliktów

Wbudowany serwer webowyTomcat

@Component – klasa jest Spring beanem

**Zasady**

1.Wstrzykiwanie zależności i rodzaje

2.Beans

3.DI + beans

4. Testowanie

**Wstrzykiwanie zależności – dependencyinjection**

Z tej techniki spring wyrósł

1. Di przez obiekt (dodatkowy param w konstruktorze)
2. Di przez metodę (przekazanie obiektu (zależności) w seterze po utworzeniu obiektu

**Spring Bean**

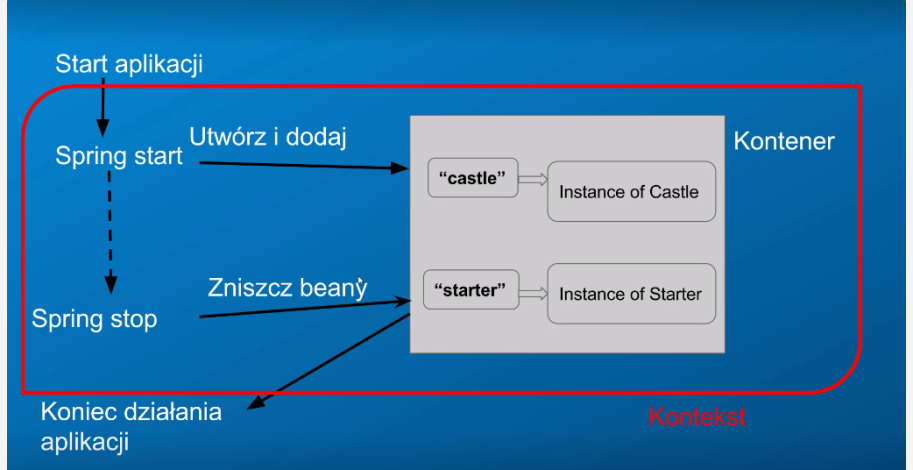
Klasy oznaczone adnotacją @Component

Spring startuje. Skanuje pliki źródłowe. Szuka np. adnotacji @Component

Utworzy obiekt tej klasy i doda do kontenera. A klasa jest dodawana do kontekstu.

**Kontekst**

Odpala metodę run na klasach implementujących CommandLineRunner



**Kontener**

to zwykła mapa, gdzie jako klucz mamy nazwę naszego beana czyli naszej klasy, a jako wartość mamy instancję naszej klasy

W momencie zakończenia aplikacji beany z kontenera są usuwane.

**Cykl życia Spring Beans**

Kontekst tworzy beany i je usuwa – to cykl ich życia.

@PostConstruct metoda uruchamia się od razu po utworzeniu beana

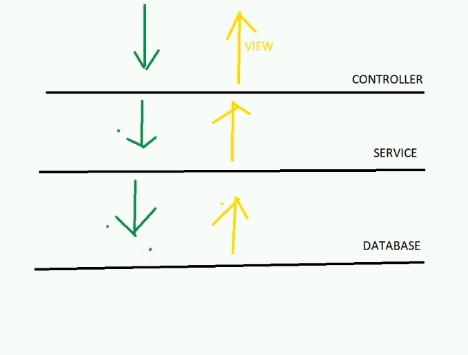
@PreDestroy metoda uruchamia się na chwilę przez usunięciem beana

// Akademia Kodu

Jest tu zasada odwrócenia sterowalności (zależności).

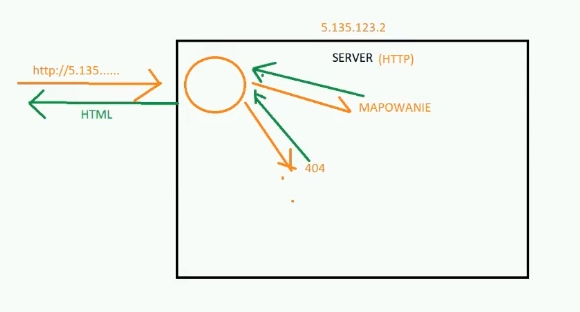
Schodzi z nas pewna odpowiedzialność. Nie będziemy już odpowiedzialny za np. przechowywanie obiektów czy wstrzykiwanie obiektów. Spring zarządza sam wszystkimi obiektami.

Warstwy. Klient uruchamia coś na stronie. Przypomina to warstwy.



REST

Wpisując cokolwiek w adresie przeglądarki, wysyła ona żądanie GET do serwera



GetMapping to to samo co RequestMapping(metod=Get, value=..)

Formularz

GET przesyła poufne dane przez url przeglądarki – kiepskie rozwiązanie

POST przesyła dane wewnątrz siebie, najlepsze do formularzy

Serwisy realizują funkcjonalność

Ziarno – element systemu, sam będzie ją tworzył - @Service

Aby Spring mógł zarządzać klasą (ziarnem) musi być konstruktor bezparametrowy

**public class** MainController {  
  
 **private** EntryService **entryService**;  
 @Autowired  
 **public** MainController(EntryService service){  
 **this**.**entryService**=service;  
 }

EntryService jest ziarnem więc można go wstrzyknąć w konstruktorze MainController. Wszystko dzięku @Autowired i mamy już zainicjalizowane pole entryService i możemy z niego korzystać, a wcale nie pisaliśmy new EntryService

@Autowired Hej tutaj, potrzebuje obiektu pewniej klasy. Wstrzyknij go tutaj!

Ważne jest, że w tym przypadku nie będzie już drugiej instancji wstrzykiwanego obiektu! (który jest oznaczony jako @Autowired)

@Autowired –jakbyśmy zrobili New COS() i istnieje tylko jedna instancja klasy COS na cały projekt

Wysyłanie danych do html