Dependency Injection su Spring

Code Academy

Programuok savo ateitį!

Kas yra Inversion of Control

Inversion of Control yra principas, nusakantis, kad klasės neturėtų kurti/konfigūruoti priklausomybių statiškai savo viduje. Tai turėtų būti vykdoma kitoje klasėje.

Vadovaujantis šiuo principu, priklausomybė nuo kitų objektų kūrimo iškeliama iš klasės, kur jos yra naudojamos, į tam specialiai deleguotą klasę.

Code Academy
Programuok savo ateitį!

Dependency Injection

Pagal teisingus programavimo principus, klasė turėtų būti koncentruota į savo įsipareigojimų vykdymą, o ne į objektų, reikalingų tų įsipareigojimų vykdymui, kūrimą.

Tam pasitelkiama *Dependency Injection*. Tai veiksmas, kai klasei pateikiami visi jai reikalingi objektai iš tam įgaliotos klasės. Tokiu būdu objektų kūrimas nevyksta klasės viduje.

Code Academy
Programuok savo ateitį

Dependency Injection

Klasės, kurių priklausomybės yra valdomos naudojant *Dependency Injection* paprastai vadinamos **komponentais**. Sistemos, naudojančios *Dependency Injection* principą deleguoja klasę, paprastai vadinamą **konteineriu (IoC konteineriu)** arba **kontekstu**, kuris atsakingas už komponento objektų inicializavimą.

Code Academy
Programuok savo ateitį

Dependency Injection privalumai

- Palengvina *unit* testų rašymą
- Sumažina *boilerplate* kodo kiekį, kadangi priklausomybių inicializavimas vykdomas tam skirtame komponente
- Palengvina sistemos plečiamumą
- Sumažina klasių tarpusavio sąryšį

Code Academy
Programuok savo ateitį!

Dependency Injection su Spring

Klasės, už kurių objektų inicializavimą bei gyvavimą yra atsakingas Spring karkasas, vadinamos **Spring beans**.

BeanFactory - sąsaja, pateikianti funkcijas, kurios gali kurti ir valdyti bet kokio tipo objektus kaip Spring beans.

Spring beans objektai yra saugomi **Spring IoC konteineryje** ir kartu su Bean Factory sudaro Spring IoC konteinerio pagrindą

Code Academy
Programuok savo ateitį!

Dependency Injection su Spring

ApplicationContext - sąsaja, praplečianti BeanFactory *enterprise-specific* funkcionalumu. Dirbant su Spring dažniausiai yra naudojama ApplicationContext, o ne BeanFactory sąsaja.

org.springframework.context.ApplicationContext sąsaja reprezentuoja Spring IoC konteinerį, kuris yra atsakingas už Spring bean objektų kūrimą, konfigūravimą, valdymą. Konteineris gauna instrukcijas, kaip ir kokius objektus inicializuoti iš konfigūravimo meta informacijos. Meta informacija gali būti pateikta kaip XML, Java anotacijos, Java programavimo kodas.

Code Academy
Programuok savo ateitį!

Dependency Injection su Spring

Klasės, turinčios anotacijas *@Component, @Service, @Controller, @Repository* yra laikomos Spring beans.

Šių klasių inicializavimui Spring karkasas naudoja anotaciją @Component*Scan* - surasti Spring komponentams ir anotaciją *@Autowired* - pateikti reikalingoms priklausomybėms.

Spring sistemos paleidimo metu vyksta Spring komponentų paieška - pagrindinė klasė yra paženklinama *@ComponentScan* anotacija. Visi paketai, esantys pagrindinės klasės paketo viduje, yra skanuojami ir rasti Spring Beans yra inicializuojami pateikiant priklausomybės paženklintas *@Autowired* anotacija.

Code Academy
Programuok savo ateitį

Dependency Injection su Spring pavyzdys

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class DatabaseAccountService implements AccountService {
	private final RiskAssessor riskAssessor;

	@Autowired
	public DatabaseAccountService(RiskAssessor riskAssessor) {
		this.riskAssessor = riskAssessor;
	}
	// ...
}
```

Code Academy
Programuok savo ateitį

@Autowired anotacijos naudojimo budai

Nurodyti priklausomybes galima trimis būdais:

- rašant anotaciją ant konstruktoriaus
- rašant anotaciją ant klasės lauko
- rašant anotaciją ant *setter* metodo

Rekomenduojama naudoti anotaciją virš konstruktoriaus - tai suteikia galimybę paduoti priklausomybes ne per iš Spring konteksto, testų metu.

Code Academy
Programuok savo ateitį!

@Bean anotacija

Klasėms, kurios nėra mūsų kodo dalis, negalima užrašyti *@Component* anotacijos. Kaip jas aprašyti kaip Spring beans, tvarkomus Spring konteinerio?

Tam naudojamos anotacijos *@Configuration* ir *@Bean*. Sukuriama klasė turinti *@Configuration* anotaciją - ši anotacija nusako, kad joje bus aprašomi Spring komponentai.

Klasėje aprašomi metodai, kurie inicializuoja ir grąžina mums reikalingas priklausomybes. Šie metodai paženklinami *@Bean* anotacija.

Code Academy
Programuok savo ateitį

@Bean anotacijos pavyzdys

```
@Configuration
public class AppConfiguration {

	@Bean
	public FirstDependency getFirstDependency() {
		return new FirstDependency();
	}

	@Bean
	public SecondDependency getSecondDependency(FirstDependency dependency) {
		return new SecondDependency(dependency);
	}
}
```

Code Academy
Programuok savo ateitį

Spring beans apimtys

Spring komponentai turi skirtingas apimtis (*scope*)

- *Singleton* - egzistuoja tik vienas klasės objektas per visą sistemą. Visur, kur prašoma, paduodama ta pati objekto instancija. **Tai yra visų komponentų apimtis, pagal nutylėjimą**
- *Prototype* - kiekvieną kartą paduodant priklausomybę su šia apimtimi yra inicializuojamas naujas objektas
- *Request* - kiekvieną kartą paduodant priklausomybę su šia apimtimi tos pačios HTTP užklausos apimtyje, yra naudojama ta pati objekto instancija.
- *Session* - kiekvieną kartą paduodant priklausomybę su šia apimtimi tos pačios sesijos apimtyje, yra naudojama ta pati objekto instancija.

Code Academy
Programuok savo ateitį!

@SpringBootApplication anotacija

Norint pasinaudoti *Spring boot* siūlomomis funkcijomis, pagrindinė sistemos klasė turėtų būti paženklinta *@SpringBootApplication* anotacija. Ši anotacija įjungia:

- ***@EnableAutoConfiguration*** - įjungia Spring Boot auto konfigūracijos mechanizmą
- ***@ComponentScan*** - įjungia komponentų skenavimą nuo paketo, kuriame aprašyta klasė su šia anotacija.
- ***@Configuration*** - leidžia registruoti papildomus Spring beans šioje klasėje

@SpringBootApplication anotacija yra ekvivalenti *@Configuration, @EnableAutoConfiguration* ir *@ComponentScan* anotacijoms

Code Academy
Programuok savo ateitį

Užduotis

Sugalvokite sistemą, kuriai realizuoti reikėtų *service* ir *repository* lygmenų.

Repozitorijos lygmuo turi dvi operacijas: sistemos esybių įrašymą į failą, ir visų jų nuskaitymą iš failo.
Serviso lygmenyje vyksta repozitorijos duomenų apdirbimas pagal sistemos reikalavimus.

Sukurkite *Spring* programėlę, kurioje serviso ir repozitorijos komponentai būtų įgyvendinti kaip *Spring beans*, o sistemos seka aprašyta Spring command runner bibliotekos pagalba.

Code Academy
Programuok savo ateitį

Naudingos nuorodos

- https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html