Dependency Injection in Spring

SOLID

- S - Single responsibility principle
- O - Open/closed principle
- L - Liskov substitution principle
- I - Interface segregation principle
- D - Dependency inversion principle

https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)

Принцип інверсії залежностей

Принцип формулюється наступним чином:

- *Модулі вищого рівня не повинні залежати від модулів нижчого рівня. Обидва типи модулів повинні залежати від абстракцій.*
- *Абстракції не повинні залежати від деталей реалізації. Деталі реалізації повинні залежати від абстракцій.*

Щоб уникнути залежності іншої від абстракцій потрібно:

- Усі змінні-члени класу повинні бути або абстрактними або інтерфейсами;
- Класи не повинні успадковуватись від не абстрактних класів;
- Методи не повинні заміщати методи реалізації;
- Усі модулі класів повинні приєднуватись через модулі інтерфейсів/абстрактних класів;
- Усі об'єкти повинні створюватись через твірні шаблони: фабричний метод чи фабрику чи через бібліотеки впровадження залежностей.

Spring IoC контейнер

- Всі біни живуть тут
- IoC керує життєвим циклом біна
- Як наша програма може дізнатися про існування біна?
- Область видимості біна

Типи зв'язування

- 1. В поле
- 2. Через мутатори
- 3. За допомогою конструтора
- Lombock
- @Qualified
- @Component, @Service, @Repository

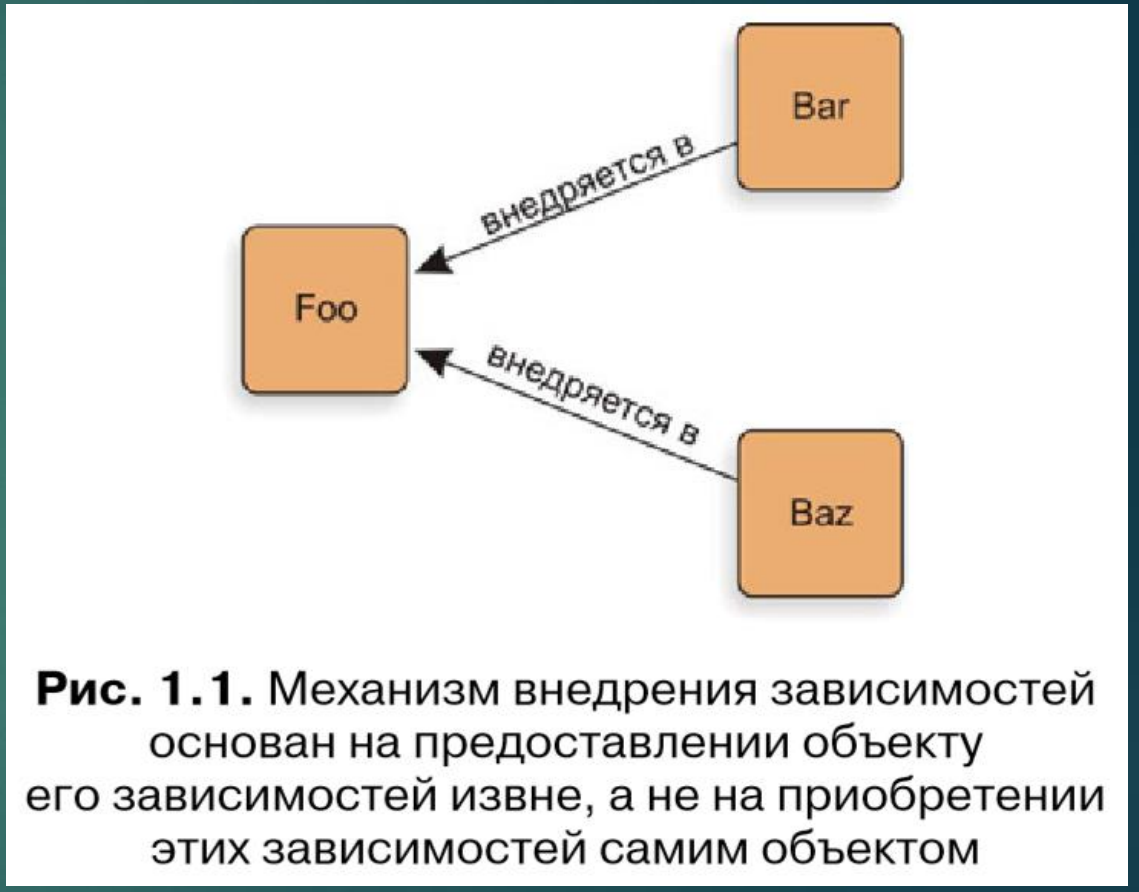

Рис. 1.1. Механизм внедрения зависимостей основан на предоставлении объекту его зависимостей извне, а не на приобретении этих зависимостей самим объектом

http://vojtechruzicka.com/field-dependency-injection-considered-harmful/

Bad code example

Листинг 1.3. Класс DamselRescuingKnight может принимать только экземпляр класса RescueDamselQuests

```
package com.springinaction.knights;

public class DamselRescuingKnight implements Knight {
     private RescueDamselQuest quest;

     public DamselRescuingKnight() {
          quest = new RescueDamselQuest();  // Тесная связь с классом
     }                                      // RescueDamselQuest

     public void embarkOnQuest() throws QuestException {
          quest.embark();
     }
}
```

Good code example

Листинг 1.4. Класс BraveKnight, достаточно гибкий, чтобы совершить любой подвиг

```
package com.springinaction.knights;

public class BraveKnight implements Knight {

private Quest quest;

public BraveKnight(Quest quest) {
     this.quest = quest;                          // Внедрение сценария подвига
}

public void embarkOnQuest() throws QuestException {
     quest.embark();
}
```

Testing

- Завжди тестуйте свої проекти
- І приклад з @Qualifier

Что почитать?

- 1. Spring in Action, Craig Walls
- 2. Spring docs: https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html
- 3. http://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-spring-beans-and-dependency-injection.html