Lesson 31

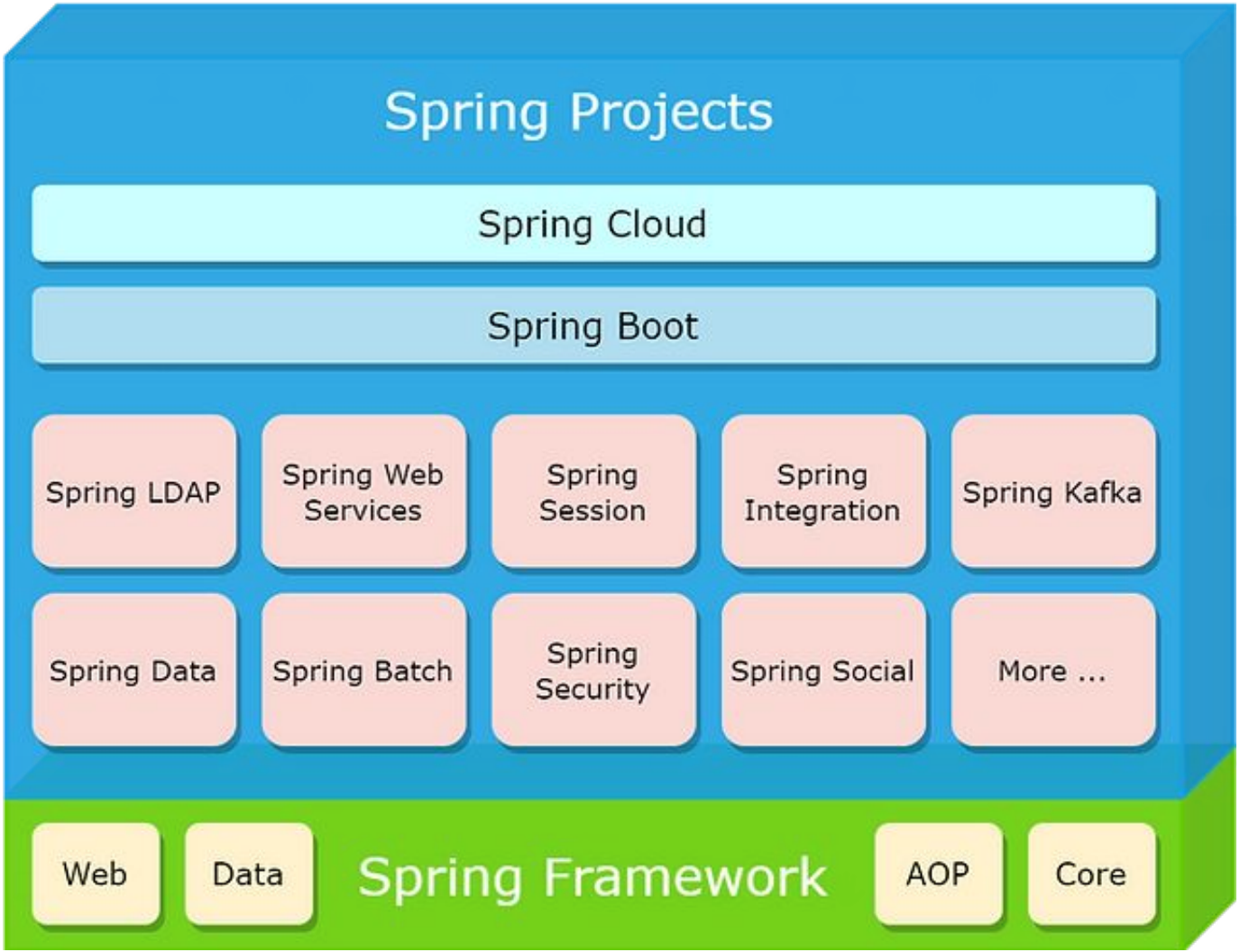
04.09.2023

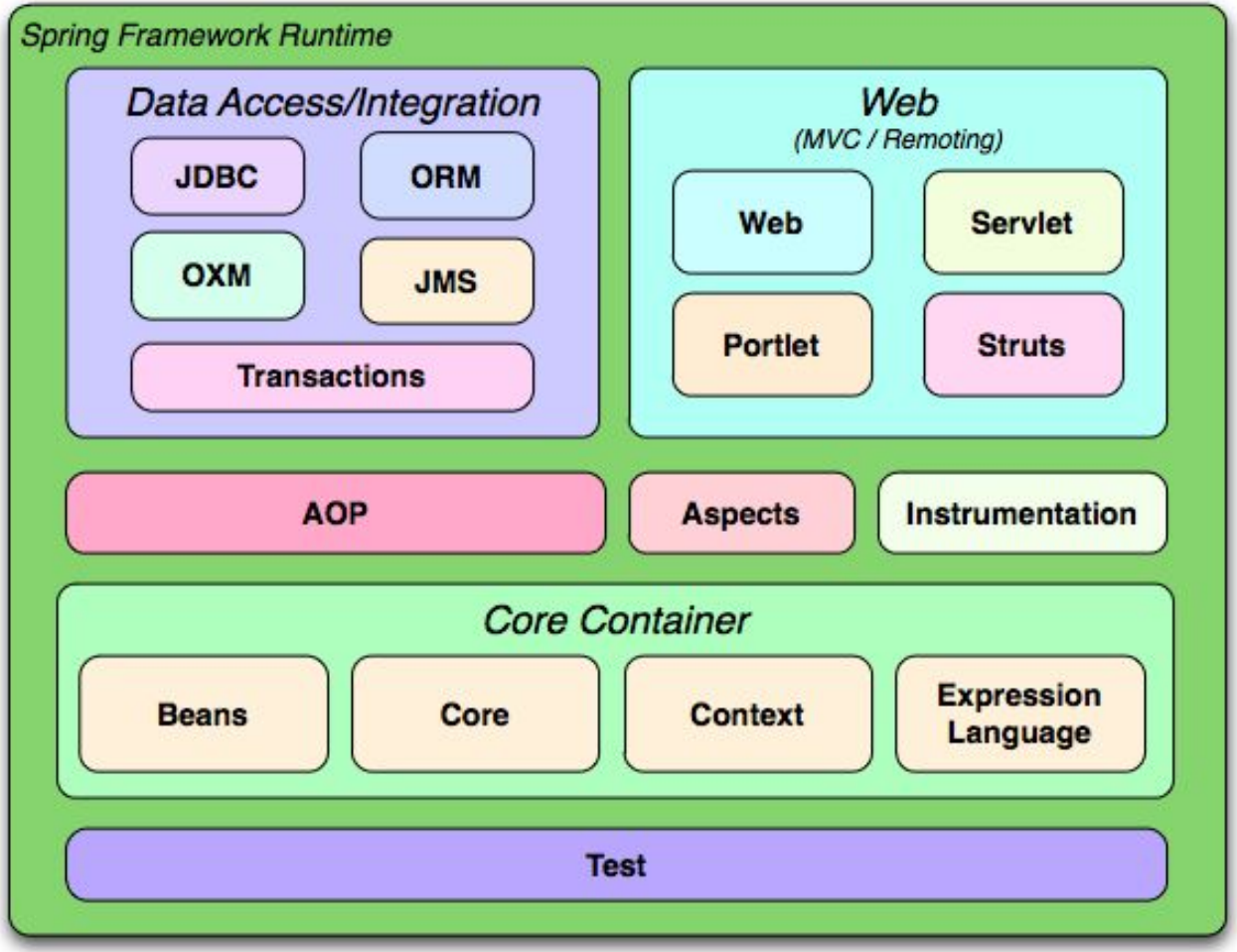
```
public class Test1 {
    static boolean foo (char c) {
        System.out.println(c);
        return true;
    }

    public static void main(String[] args) {
        int i = 0;
        for (foo('A'); foo('B') && (i < 3); foo('C')) {
            i++;
            foo('D');
        }
    }
}
```

```
public class Test2 {
    public static void main(String[] args) {
        Boolean[] boo = new Boolean[3];
        boo[0] = new Boolean(Boolean.parseBoolean("tRUE"));
        boo[1] = new Boolean("True");
        boo[2] = new Boolean(false);
    }
}
```

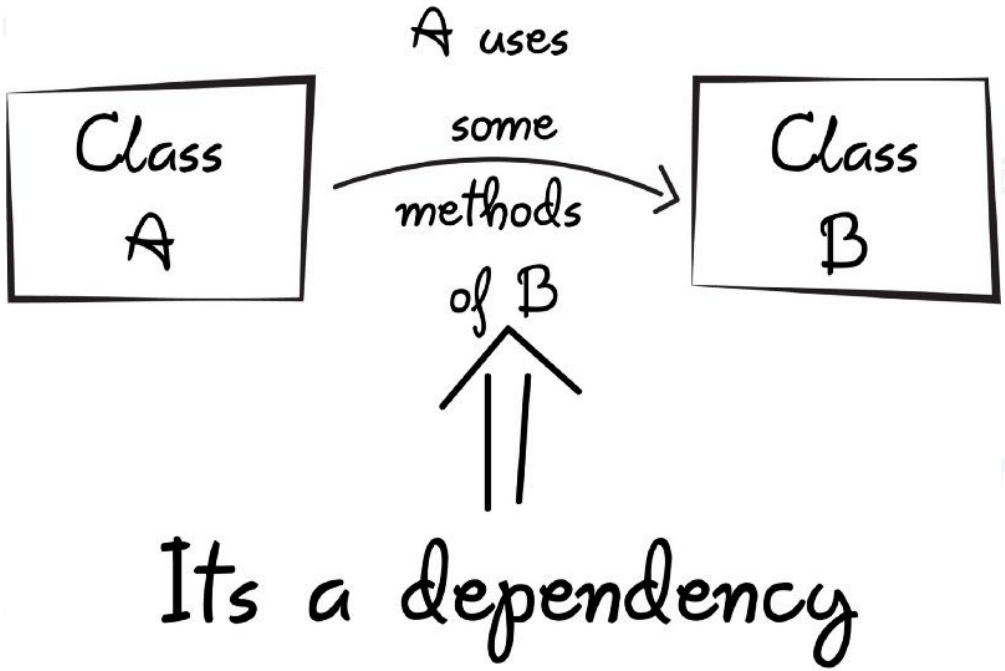
Version	Date
0.9	2003
1.0	March 24, 2004
2.0	2006
3.0	2009
4.0	2013
5.0	2017
6.0	November 16, 2022


Spring Projects
Spring Cloud
Spring Boot
Spring LDAP
Spring Web Services
Spring Session
Spring Integration
Spring Kafka
Spring Data
Spring Batch
Spring Security
Spring Social
More ...
Web
Data
Spring Framework
AOP
Core


Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
(MVC / Remoting)
Web
Servlet
Portlet
Struts
AOP
Aspects
Instrumentation
Core Container
Beans
Core
Context
Expression Language
Test

У розробці програмного забезпечення, використання залежностей це така техніка, де за допомогою одного об'єкта (або статичного методу) надаються залежності іншого об'єкта. Залежність — це об'єкт, який можна використовувати (як сервіс).

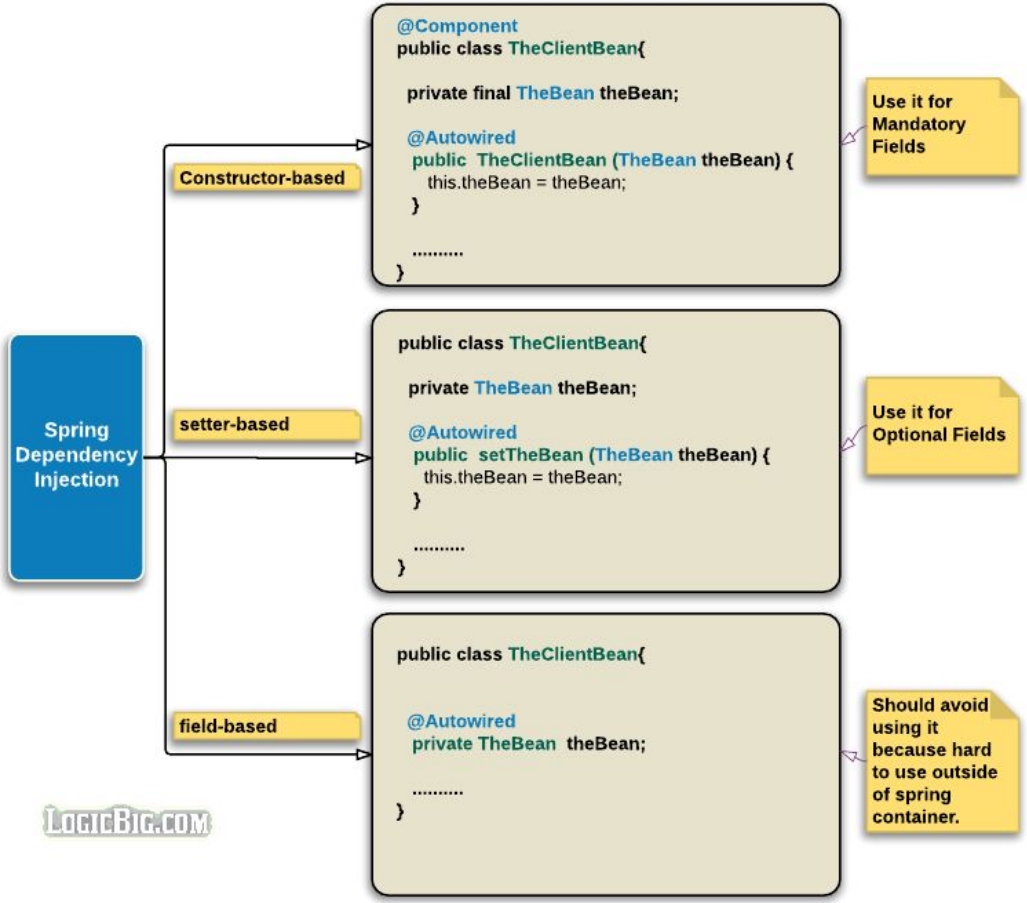
Коли клас A використовує деяку функціональність класу B, тоді кажуть, що клас A залежить від класу B.



У Java, перш ніж ми зможемо використовувати методи інших класів, нам необхідно для початку створити екземпляри цього класу (тобто, клас A повинен створити екземпляр класу B).

Таким чином, передаючи завдання створення об'єкта чомусь іншому та пряме використання цієї залежності називається впровадженням залежностей.

Different ways of DI in Spring



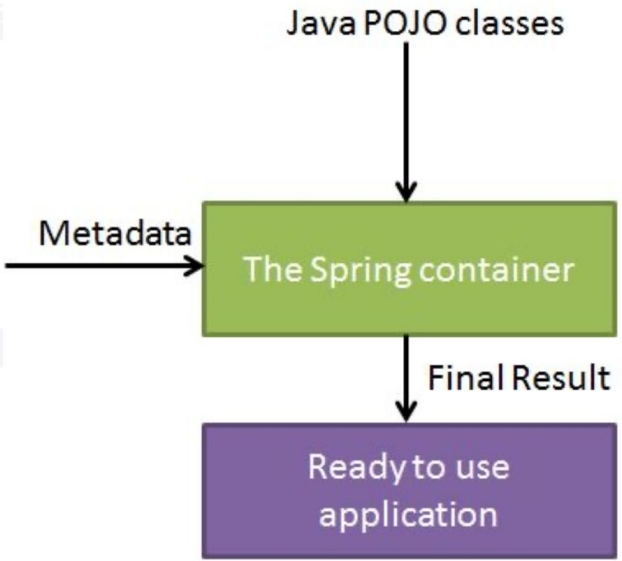
Контейнери IoC

Ключовим елементом Spring Framework є Spring Container. Container створює об'єкти, пов'язує їх разом, налаштовує та керує ними від створення до моменту знищення.

Для керування компонентами, з яких складається програма, Spring Container використовує Впровадження Залежностей (DI). Ці об'єкти називаються Spring Beans, які ми обговоримо далі. Spring Container отримує інструкції, які об'єкти інстанціювати і як їх конфігурувати через метадані.

Метадані можуть бути отримані 3 способами:

- XML
- Анотації Java
- Java код



На малюнку, схематично показаний цей процес. Java POJO класи надходить у Spring Container, який на підставі інструкцій, отриманих через метадані, видає програму, готову до використання.

У Spring є 2 різних види контейнерів:

1. Spring BeanFactory Container;
2. Spring ApplicationContext Container;

Spring ApplicationContext Container

ApplicationContext завантажує біни, зв'язує їх разом і конфігурує їх певним чином. Але крім цього, ApplicationContext має додаткову функціональність: розпізнавання текстових повідомлень з файлів налаштування та відображення подій, які відбуваються в додатку різними способами. Цей контейнер визначається інтерфейсом org.springframework.context.ApplicationContext.

Найчастіше використовуються наступні реалізації AppicationContext:

- FileSystemXmlApplicationContext
– ClassPathXmlApplicationContext
– WebXmlApplicationContext

Bean

Біни – це об'єкти, які є основою програми та управляються Spring IoC контейнером. Ці об'єкти створюються за допомогою конфігураційних метаданих, які вказуються в контейнері (наприклад, XML-<bean>…</bean>). Я вже говорив про них у попередніх розділах.

Визначення бина містить метадані конфігурації, які необхідні керуючому контейнеру для отримання наступної інформації:

Як створити бін;

Інформацію про життєвий цикл біну;

Залежність бина

class
Цей атрибут є обов'язковим і вказує конкретний клас Java-додатка, який буде використовуватися для створення бина.

name
Унікальний ідентифікатор біну. У разі конфігурації за допомогою файлу xml, ви можете використовувати властивість “id” та/або “name” для ідентифікації біна.

scope
Ця властивість визначає область видимості створюваних об'єктів. (

constructor-arg
Визначає конструктор, що використовується для впровадження залежності.

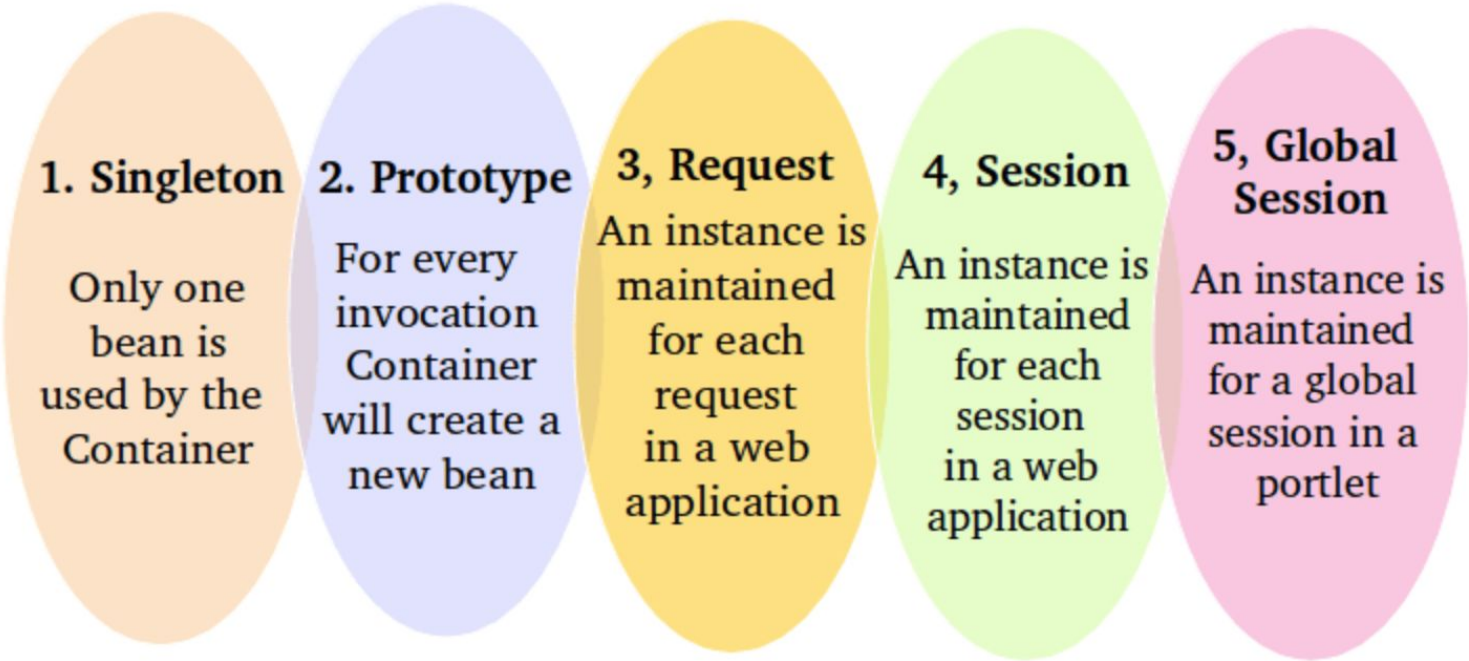
properties
Визначає характеристики застосування залежності.

initialization method
Тут визначається метод ініціалізації бина

destruction method
Метод знищення бина, який використовуватиметься при знищенні контейнера, що містить бін.

autowiring mode
Визначає режим автоматичного зв'язування під час впровадження залежності.

lazy-initialization mode
Режим лінивої ініціалізації дає контейнеру IoC команду створювати екземпляр бина при першому запиті, а не при запуску програми



- application
- WebSocket
- global session

Bean life cycle in Java Spring

Життєвий цикл будь-якого об’єкта означає, коли і як він народжується, як поводиться протягом свого життя, а також коли і як помирає. Подібним чином, життєвий цикл bean-компонента стосується того, коли і як створюється екземпляр bean-компонента, які дії він виконує, поки не залишиться живим, і коли і як його знищують. У цій статті ми обговоримо життєвий цикл біна.

Життєвим циклом бінів керує spring контейнер. Коли ми запускаємо програму, то, перш за все, запускається контейнер spring. Після цього контейнер створює екземпляр bean-компонента відповідно до запиту, а потім впроваджуються залежності. І, нарешті руйнуються, коли spring контейнер закритий. Таким чином, якщо ми хочемо виконати якийсь код на екземплярі bean-компонента і відразу після закриття контейнера spring, тоді ми можемо записати цей код у користувацькі методи init() і destroy().