# КОНЦЕПЦІЯ JAVA BEANS

# Java Bean

У документації від Sun бін визначається так: "A **Java Bean is a reusable software component that can be manipulated visually in a builder tool.**" ( "Java Bean це багаторазово використовуваний програмний компонент, яким можна маніпулювати візуально в (візуальних) середовищах розробки").

# Java Bean

У найпростішому випадку бін - це окремий клас, що представляє певну компоненту. У більш складних випадках - це набір взаємопов'язаних класів, кожен з яких відіграє певну роль. Так багато класів стандартної бібліотеки Java є бінами, наприклад, JLabel, JTextField і ін.

# Java Bean

Основний клас біна повинен задовольняти одній вимозі - він повинен мати **конструктор за замовчуванням** (default constructor). Ця вимога природно. Передбачається, що візуальне середовище буде створювати екземпляри бінов і використовувати для цього конструктори за умовчанням. Є й інші вимоги до бінів.

Біни можуть бути абсолютно різними як за розмірами, складності, так і за областю застосування. Кожен конкретний бін може підтримувати той чи інший ступінь функціональності, але він повинен забезпечувати типові універсальні можливості.

# Java Bean:

- Підтримує "інтроспекцію" (introspection), що дозволяє середовищам розробки аналізувати з чого складається і як працює даний бін.
- Забезпечує зручні налаштування (customization), тобто можливість змінювати зовнішній вигляд (положення, розміри і т.п.) і поведінку.
- Забезпечує підтримку "подій" (events) як засобу зв'язку даного біна з програмою і іншими бінами.
- Забезпечує підтримку властивостей або атрибутів (properties), які використовуються, зокрема, для настройки (наприклад, ширина, висота, кількість яких-небудь складових ).
- Підтримує "збереженість" (persistence). Це необхідно для того, щоб після настройки конкретного біна була можливість зберегти параметри налаштування, а потім їх відновити.

# Enterprise Java Beans

EJB - специфікація, надана Sun Microsystems для розробки захищених, надійних та масштабованих розподілених додатків.

Для запуску програми EJB вам потрібен сервер додатків, такий як Jboss, Glassfish, Weblogic, Websphere тощо.

Він виконує:

управління життєвим циклом,

безпека,

управління транзакціями та

об'єднання об'єктів.

Програма EJB розгортається на сервері, тому її також називають компонентом на стороні сервера.

EJB є аналогом COM Майкрософт.

# Enterprise Java Beans

EJB – використовують, якщо:

Додаток потребує віддаленого доступу.

Додаток має бути масштабованим . Програми EJB підтримують балансування навантаження, кластеризацію.

Додаток потребує інкапсульованої бізнес-логіки.

У Java існує 3 типи enterprise bean in java.

Session Bean містить бізнес-логіку, яку може викликати локальний, віддалений або клієнт веб-сервісу.

Message Driven Bean містить бізнес-логіку, але викликається передачею повідомлення.

Entity Bean інкапсулює стан, який може зберігатися в базі даних. Замінений на JPA.

# Фреймворк Spring

Spring - це вільно поширюваний фреймворк, створений Родом Джонсоном (Rod Johnson) і описаний в його книзі «Expert One-on-One: J2EE Design and Development». Він був створений з метою усунути складнощі розробки корпоративних додатків і зробити можливим використання простих компонентів JavaBean для досягнення всього того, що раніше було можливим тільки з використанням EJB.

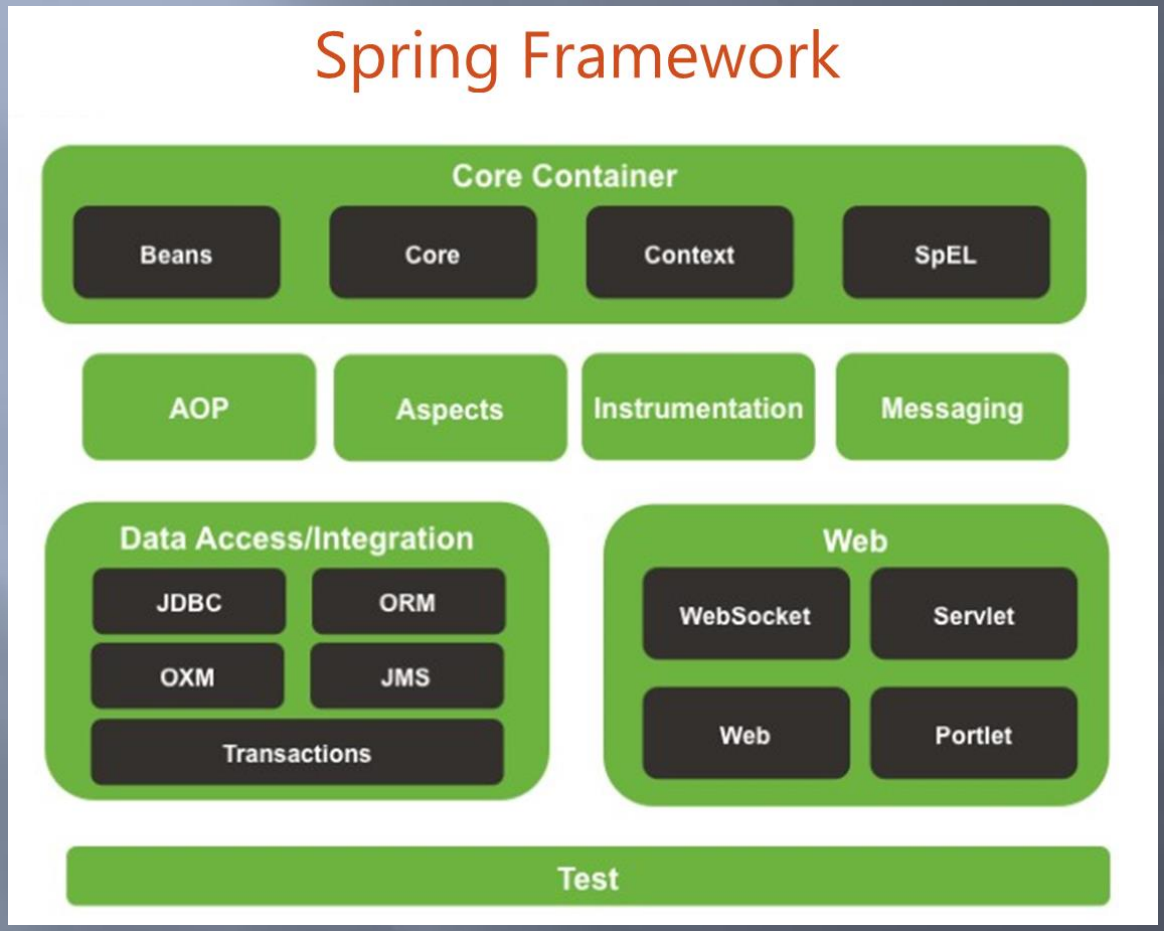
Розпочався з двох базових ідей: **Dependency Injection** та AOP (Aspect Oriented Programming)

# Базові підходи Spring

- **Dependency Injection (DI) -** технологія, з якою найбільше ототожнюється Spring, - це основа для інверсії контролю (**Inversion of Control (IoC)** ) – підходу до створення і керування життєвим циклом об'єктів за допомогою зовнішніх контейнерів. Такий підхід виник, як узагальнення шаблону проектування Абстрактна фабрика і застосовується до розробки масштабовних систем, тестування і т. п.
- **Aspect Oriented Programming (AOP)** Модуль AOP Spring Framework забезпечує реалізацію програмно-орієнтованих наскрізних аспектів, що дозволяє визначити перехоплювачі методів і точки для чистого роз'єднання коду, який реалізує різнопланові функціональні можливості.

# Історія версій Spring

- Перша версія була написана Родом Джонсоном, який випустив фреймворк з публікацією його книги Expert One-on-One J2EE Design and Development в жовтні 2002 р.
- Фреймворк був вперше випущений за ліцензією Apache 2.0 у червні 2003 р. випуск 1.0 був випущений в березні 2004 р.
- Spring 2.0 жовтень 2006 р., Spring 2.5 листопад 2007, Spring 3.0 грудень 2009 року, Spring 3.1 грудень 2011 року та Spring 3.2.5 у листопаді 2013
- Spring Framework 4.0 був випущений в грудні 2013 р. Помітні вдосконалення у Spring 4.0 включали підтримку Java SE (Standard Edition) 8, Groovy 2, деякі аспекти Java EE 7 та WebSocket .
- Spring 5, що вийшов в 2017 побудований на реакторному ядрі, сумісному з реактивними потоками.
- Spring 5.3 ( 27 Oct 2020) поточна збірка 5.3.34 (11 Apr 2024)
- Spring 6.0 (16 Nov 2022), 6.1 (Nov 16, 2023) найновіша збірка 6.1.6 (Apr 11, 2024)




# архітектура Spring

# Spring Context

interface ApplicationContext

ClassPathXmlApplicationContext

FileSystemXmlApplicationContext

GenericGroovyApplicationContext

AnnotationConfigApplicationContext

StaticApplicationContext

# конфігурація бінів в Spring xml-context

Тег <bean>  атрибути:

- class
- name
- scope (singleton, prototype, request...)
- constructor-arg
- property (атрибути name +  value, або name + ref)
- autowiring
- lazy-init
- init-method
- destroy-method

# Списки в Spring xml-context

```
<bean id = "javaCollection" class
="com.tutorialspoint.JavaCollection">
        <property name = "addressList">
            <list>
                <value>INDIA</value>
                <value>Pakistan</value>
                <value>USA</value>
                <value>USA</value>
            </list>
        </property>
    </bean>
```

# Анотації Spring

1. Конфігурація контейнера на основі анотацій spring @Component, @Bean, @Scope, @Configuration, @ComponentScan.

2 Spring підтримує анотації на основі JSR-250, які включають анотації @Resource, @PostConstruct та @PreDestroy.

3 @Autowired - анотація може застосовуватися до методів встановлення властивостей компонента, методів, що не задають задачу, конструктора та властивостей. @Autowired (обов'язково = "false") - не обовязкове до заповнення.

4 @Required - поле вимагає заповнення при старті контейнера.

3 @Qualifier - анотація разом із @Autowired може бути 5икористана для усунення плутанини, вказавши, який саме компонент буде підключений.