1.1 Introduction to Spring Framework and IOC Container

Why do we need Spring Framework over Java EE?

Before **2003**, developers used **Java EE (Jakarta EE)** to build *enterprise applications*, but it was **complex, heavy**, and required **lengthy configuration** before writing any *business logic*.

👨‍💻 **Problem:**

- Too much boilerplate code.
- Complex **XML** configuration.
- **Tight coupling** between components.
- Difficult to **test** and **maintain**.

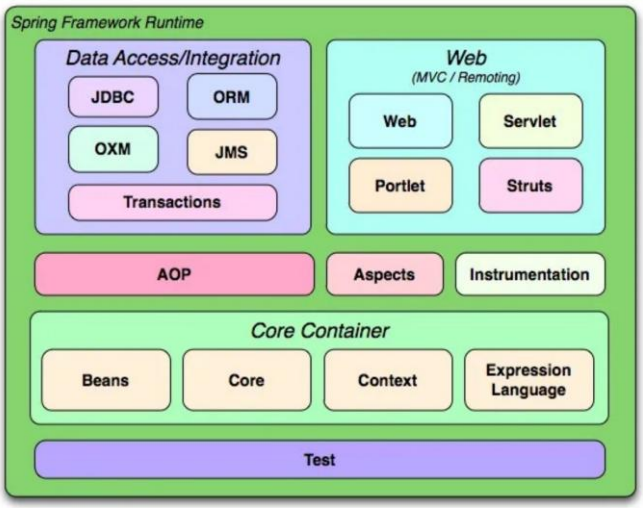
🌱 **Spring Framework Solution:** Introduced by **Rod Johnson in 2003**, Spring provided a **lightweight**, **modular**, and **non-invasive** approach using:

- **Dependency Injection (DI)** for loose coupling.
- **Aspect-Oriented Programming (AOP)** for separating concerns.
- **Easy integration** with existing tech (Hibernate, JDBC, etc.).
- **Minimal configuration** (later further simplified by Spring Boot).

While **Spring Framework** made Java development *easier* than **Java EE**, it still required **a lot of configurations** to get started— setting up **beans**, **servers**, and **dependencies** manually.
To solve this, in **2014**, after **VMware** acquired Spring, they introduced **Spring Boot**.

What is Spring Framework?

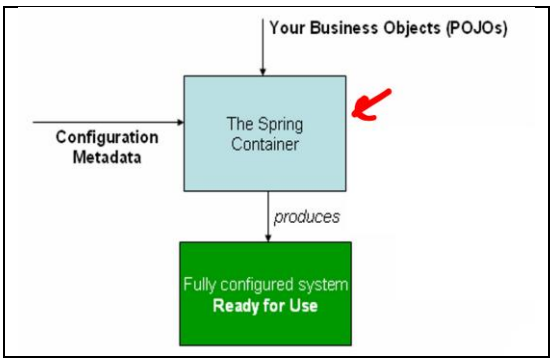
- **Spring** is a **Dependency Injection framework** to make java applications **loosely coupled**.
- **Spring framework** makes the *development process* easy for **JavaEE** applications.
- Spring enables you to build applications from "**plain old Java objects**" (POJOs) and to apply enterprise services non-invasively to **POJOs**.
- **Spring** was developed by **Rod Johnson** in **2003.**



Important Components:

- Core Container
- AOP (Aspect-Oriented Programming)
- JDBC (Java Database Connectivity)
- Web
- Testing

What is IOC container?



In the **Spring Framework**, the IoC container is responsible for **managing** the **components** of an application and **injecting dependencies** into them. The container creates the **objects** (**beans**), **wires** them together, **configures** them, and manages their **complete lifecycle**.