# Spring - IoC Container

Last Updated : 23 Jul, 2025

The **Spring framework** is a powerful framework for building Java applications. It can be considered a collection of sub-frameworks, also referred to as layers, such as **Spring AOP, Spring ORM, Spring Web Flow, and Spring Web MVC**. We can use any of these modules separately while constructing a Web application. The modules may also be grouped to provide better functionalities in a web application.

## Spring Container

The core component of the spring framework is the Inversion of Control (IOC) container, which is responsible for managing the lifecycle of objects called **beans** and their dependencies. Spring provides two types of containers:

- **BeanFactory Container**
  - This is the basic container that provides support for dependency injection.
  - In this, beans are instantiated only when explicitly requested.
  - It is lightweight and suitable for resource-constrained environments.
- **ApplicationContext Container**
  - This is an advanced container built on top of the BeanFactory.
  - It includes all the features of BeanFactory and adds extra functionalities such as internationalization, event propagation, and integration with other Spring modules.
  - In this, beans are created and configured at startup.

### Key feature of Spring Framework

The features of the Spring framework, such as IoC, AOP, and transaction management, make it unique among Java frameworks. Some of the most important features include:

- IoC Container
- Data Access Framework (JDBC abstraction layer)
- Spring MVC
- Transaction Management
- Spring Web Services
- Spring TestContext Framework

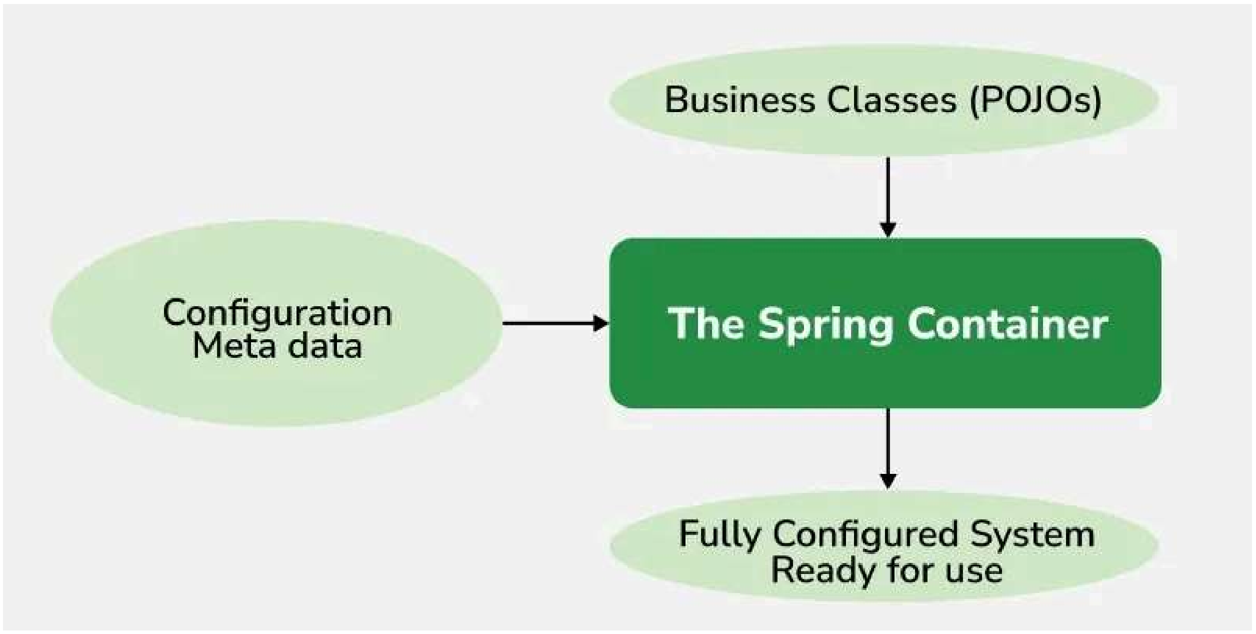
## Spring IOC Container

Spring IoC is a design principle where the control of object creation and lifecycle is transferred from the developer to the framework. The IoC container is responsible for:

1. Creating objects (Beans)
2. Configuring dependencies via Dependency Injection (DI)
3. Managing the entire lifecycle of beans, from instantiation to destruction
4. Reading configuration metadata (XML, Java Config, or Annotations)

> ***Note:*** *The objects managed by the container are called* ***beans.***

The below diagram demonstrates how the Container makes use of Configuration metadata and Java POJO classes to manage beans.



## Working of IoC Container

- The container reads configuration metadata (XML, Java annotations, or Java-based configuration) to understand how beans should be creates and wired together.
- It uses Dependency Injection (DI) to inject dependencies into beans at runtime.
- The container manages the entire lifecycle of beans, from instantiation to destruction.

## BeanFactory vs ApplicationContext

The below table demonstrates the key difference between BeanFactory and ApplicationContext

| Feature | BeanFactory | ApplicationContext |
|---|---|---|
| **Annotation Support** | No | Yes |
| **Bean Instantiation/Wiring** | Yes | Yes |

| Feature | BeanFactory | ApplicationContext |
|---|---|---|
| **Internationalization** | No | Yes |
| **Enterprise Services** | No | Yes |
| **ApplicationEvent Publication** | No | Yes |
| **Automatic BeanPostProcessor Registration** | No | Yes |
| **Loading Mechanism** | Lazy Loading | Aggressive Loading |
| **Automatic BeanFactoryPostProcessor Registration** | No | Yes |

## Why use Spring Framework?

- **Modularity:** Use only the modules we need.
- **Flexibility**: Supports multiple configuration styles (XML, annotations, Java code).
- **Scalability:** Suitable for small applications as well as large enterprise systems.
- **Community Support:** Backed by a large and active community.