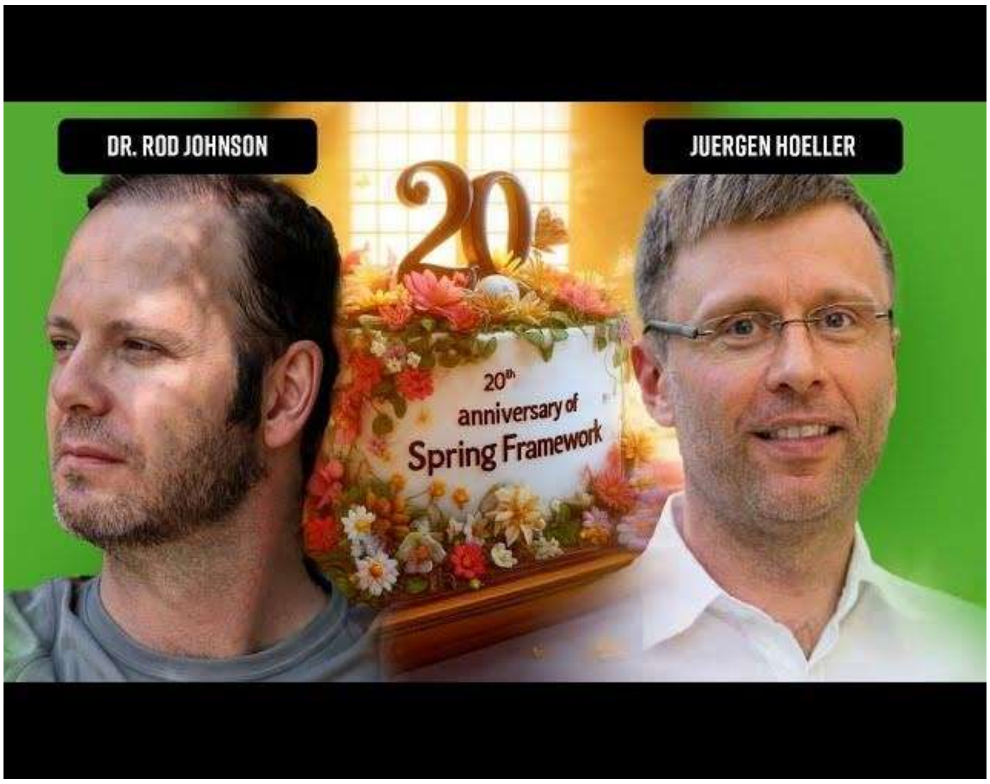
Spring Framework-Introduction

Spring is a powerful lightweight application development **framework** used for Java Enterprise Edition (JEE). In a way, it is a ***framework*** *of* ***frameworks*** because it provides support to various frameworks such as **Struts, Hibernate, Tapestry, EJB, JSF**, etc.

The **framework** in a broader sense can be defined as a structure using which you can solve many **technical problems**. You can say that the **Spring Framework** is a comprehensive tool for supporting applications using **Java programming language**.



Roderick B. Johnson, an Australian computer specialist officially released the Spring Framework in 2004. Since its origin, the Spring Framework has released many versions. **7.0.0-M6/ Jun 12, 2025** is the current Spring Framework version.

Features Of Spring Framework

- **Lightweight:** Spring Framework is lightweight with respect to size and transparency.

- **Inversion Of Control (IoC):** In Spring Framework, loose coupling is achieved using Inversion of Control. The objects give their own dependencies instead of creating or looking for dependent objects.

- **Aspect Oriented Programming (AOP):** By separating application business logic from system services, Spring Framework supports Aspect Oriented Programming and enables cohesive development.

- **Container:** Spring Framework creates and manages the life cycle and configuration of application objects.

- **MVC Framework:** Spring Framework is a MVC web application framework. This framework is configurable via interfaces and accommodates multiple view technologies.

- **Transaction Management:** For transaction management, Spring framework provides a generic abstraction layer. It is not tied to J2EE environments and it can be used in container-less environments.

- **JDBC Exception Handling:** The JDBC abstraction layer of the Spring Framework offers an exception hierarchy, which simplifies the error handling strategy.

Why do we use Spring in Java?

- Works on **POJOs (Plain Old Java Object)** which makes your application lightweight.
- Provides **predefined templates for JDBC, Hibernate, JPA** etc., thus reducing your effort of writing too much code.
- Because of **dependency injection** feature, your code becomes loosely coupled.
- Using Spring Framework, the development of **Java Enterprise Edition** (JEE) applications became faster.
- It also provides strong abstraction to Java Enterprise Edition (JEE) specifications.
- It provides declarative support for transactions, validation, caching and formatting.

But because of having all these advanced features, often people tend to ask questions like,

Is Spring easy to learn?

Java Spring is really easy to learn as the entire Spring framework is designed to work with POJOs rather than depending on special interfaces, abstract classes, etc.

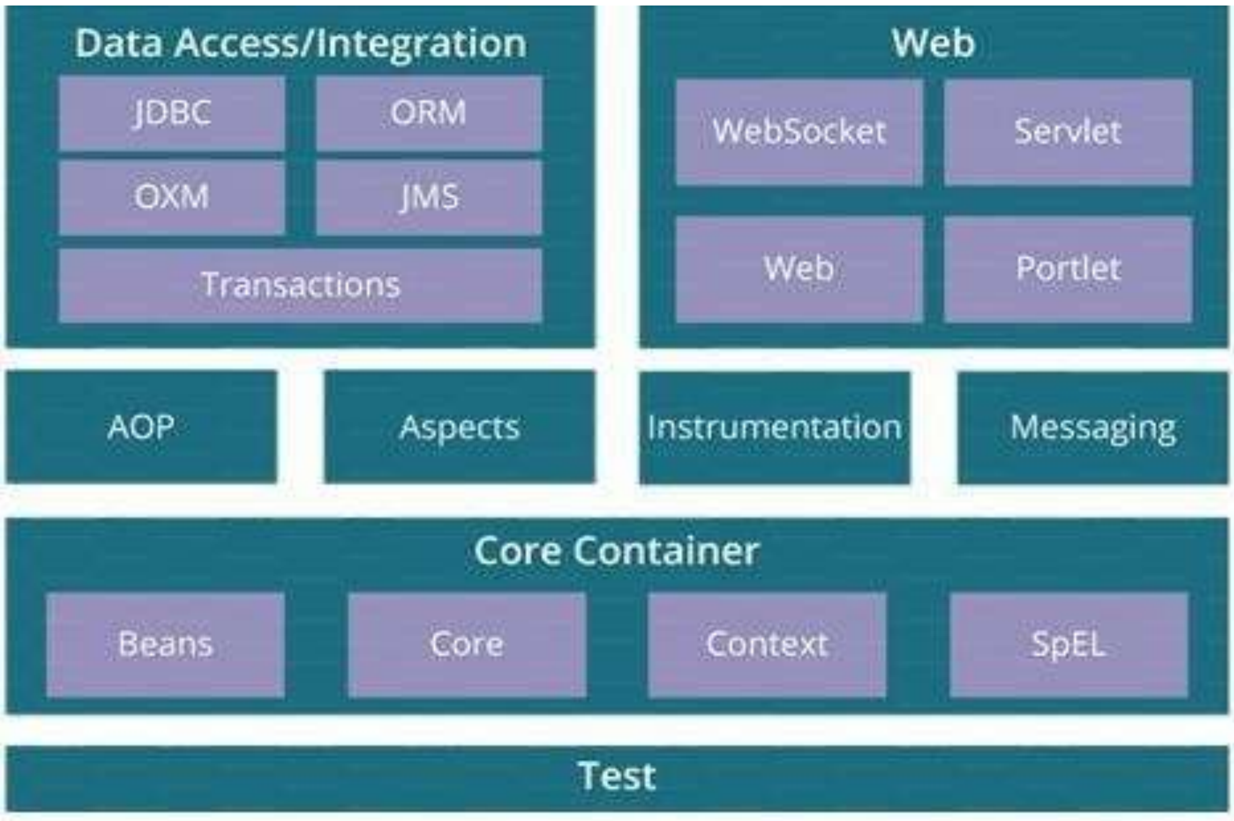
Since its origin till date, Spring has spread its popularity across various domains. Spring Framework now is the foundation for various other

Spring Framework Architecture

As you can see in the diagram below, Spring Framework architecture is an arranged layered architecture that consists of different modules. All the modules have their own functionalities that are utilized to build an application.

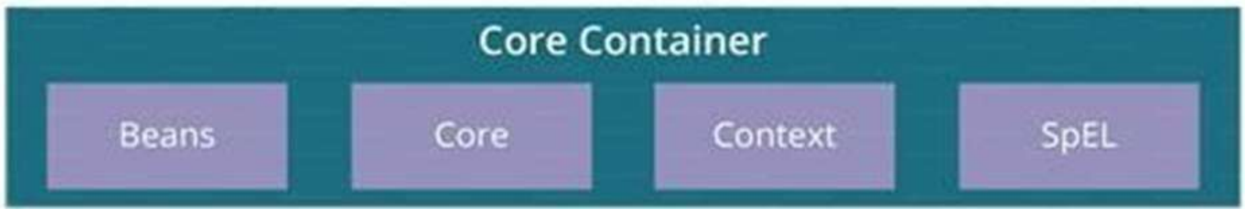
There are around **20 modules** that are generalized into **Core Container**, **Data Access/ Integration**, **Web**, **AOP** (Aspect Oriented Programming), **Instrumentation**, and **Test**.

Here, the developer is free to choose the required module. Its modular architecture enables integration with other frameworks without much hassle.



Modules of Spring Framework

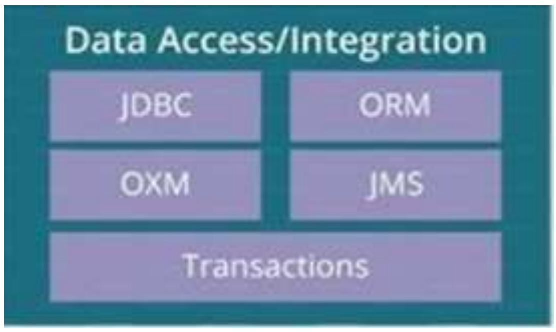
Core Container



This container has the following four modules :

1. **Spring Core:** This module is the core of the Spring Framework. It provides an implementation for features like **IoC (Inversion of Control)** and **Dependency Injection** with **a singleton** design pattern.
2. **Spring Bean:** This module provides an implementation for the factory design pattern through **BeanFactory.**
3. **Spring Context:** This module is built on the solid base provided by the Core and the Beans modules and is a medium to access any object defined and configured.
4. **Spring Expression Languages (SpEL):** This module is an extension to expression language supported by Java server pages. It provides a powerful expression language for querying and manipulating an object graph, at runtime.

Spring Data Access/ Integration



It consists of the following five modules:

1. **JDBC:** This module provides JDBC abstraction layer which eliminates the need of repetitive and unnecessary exception handling overhead.

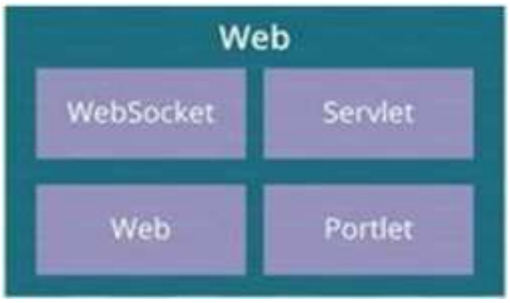
2. **ORM:** ORM stands for **O**bject **R**elational **M**apping. This module provides consistency/ portability to our code regardless of data access technologies based on object oriented mapping concept.

3. **OXM:** OXM stands for **O**bject **X**ML **M**appers. It is used to convert the objects into XML format and vice versa. The Spring OXM provides an uniform API to access any of these OXM frameworks.

4. **JMS: JMS stands for Java Messaging Service.** This module contains features for producing and consuming messages among various clients.

5. **Transaction:** This module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs. All the enterprise level transaction implementation concepts can be implemented in Spring by using this module.

Spring Web



Web layer includes the following modules:

1. **Web**: This module using servlet listeners and a web-oriented application context, provides basic web-oriented integration features like multi-part file upload functionality and the initialization of the **IoC container.**

2. **Web-Servlet**: This module contains **Model-View-Controller** (MVC) based implementation for web applications. It provides all other features of MVC, including UI tags and data validations.

3. **Web-Socket:** This module provides support for **WebSocke**t based and two- way communication between the client and the server in web applications.

4. **Web-Portlet:** This module is also known as the **Spring-MVC-Portlet** module. It provides the support for Spring-based Portlets and mirrors the functionality of a Web-Servlet module.

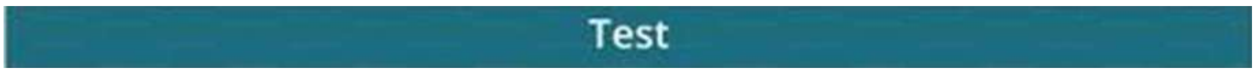
Aspect-Oriented Programming (AOP)

AOP language is a **powerful tool** that allows developers to add enterprise functionality to the application such as transaction, security etc. It allows us to write less code and separate the code logic. AOP uses **cross-cutting concerns**.

Instrumentation

This module provides class instrumentation support and classloader implementations that are used in certain application servers.

Test



This module supports the testing of Spring components with JUnit or TestNG. It provides consistent loading of **Spring ApplicationContexts** and caching of those contexts. It also provides mock objects that we can use to test our code in isolation.

In order to create your application, you might need the help of IDE (Integrated Development Environment). An IDE is a software application that provides comprehensive facilities to the programmers for software development. There are various IDE's available in the market, here I am using the **Spring Tool Suit** because it's very convenient to use.

Spring IoC Container

So what exactly is an **IoC container in Spring**? Well, Spring IoC stands for **Inversion of Control**. It is the heart of the Spring Framework. The important tasks performed by the IoC container are:

1. **Instantiating the bean**
2. **Wiring the beans together**
3. **Configuring the beans**
4. **Managing the bean's entire life-cycle**

There are **two types** of **IoC** containers. They are:

1. **BeanFactory**
2. **ApplicationContext**

Difference between BeanFactory and the ApplicationContext

The **org.springframework.beans.factory.BeanFactory** and the **org.springframework.context.ApplicationContext** interfaces acts as the **IoC** container.

The **ApplicationContext** interface is built on top of the **BeanFactory** interface. It adds some extra functionality than BeanFactory such as simple integration with Spring's AOP, message resource handling **(for I18N), event propagation, application layer specific context (e.g. WebApplicationContext) for web application.**

So it is better to use **ApplicationContext** than **BeanFactory.**

What is Dependency Injection?

Dependency Injection is the ability of an object to supply dependencies of another object.

When you hear the term dependency, what comes on to your

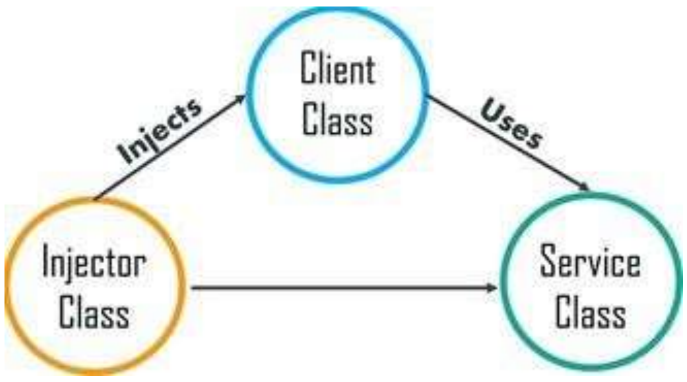
mind? Obviously, something relying on something else for support

right?

Well, that's the same, in the case of programming also.

Dependency in programming is an approach where a class uses specific functionalities of another class. So, for example, If you consider two classes A and B, and say that class A uses functionalities of class B, then its implied that class A has a dependency of class B.

Now, if you are coding in Java then you must know that, you have to create an instance of class B before the objects are being used by class A.



So, if I have to now define Dependency Injection for you, then the process of creating an object for some other class and let the class directly using the dependency is called Dependency Injection. It mainly has three classes involved:

- **Client Class:** This is the dependent class and is dependent on the Service class.
- **Service Class:** This class provides a service to the client class.
- **Injector Class:** This class is responsible for injecting the service class object into the client class

Inversion of Control

Inversion of Control is a principle based on which, Dependency Injection is made. Also, as the name suggests, Inversion of Control is basically used to invert different kinds of additional responsibilities of a class rather than the main responsibility.

Consider an example, wherein you have the ability to cook. According to the IoC principle, you can invert the control, so instead of you cooking food, you can just directly order from outside, wherein you receive food at your doorstep. Thus the process of food delivered to you at your doorstep is called the Inversion of Control.

You do not have to cook yourself; instead, you can order the food and let a delivery executive, deliver the food for you. In this way, you do not have to take care of the additional responsibilities and just focus on the main work.

Think of it this way:

Imagine you love cooking. If you do everything yourself:

- You **buy groceries**
- You **cook the food**
- You **serve yourself**
- You **wash dishes**

This is like **traditional programming**, where your class creates and manages everything itself.

Inversion of Control Analogy

Instead of cooking yourself:
You **order food** on a food delivery app.
The **restaurant cooks the food**
The **delivery executive brings it to your door**

Now you don't worry about how the cooking and delivery happens.
You just say:

"Bring me food!"

This is Inversion of Control:

- You **gave up control** of cooking.
- Someone else handles it **for you**.
- You just use the ready-made result.

Key Points to Remember :

1. **IoC flips the control**:
   Instead of *your code* creating objects, the framework creates them.
2. **You just declare what you need** (dependencies), and the framework provides them.
3. **Dependency Injection** is a common way of implementing IoC.
   - **"Injection"** means the framework *injects* the objects you need.
4. **Why use IoC?**
   Less code to write
   Easier to change parts of your program
   Easier to test
   More flexible

Quick analogy recap:

- **Traditional programming:** You cook everything yourself.
- **Inversion of Control:** You place an order, and someone else handles the cooking and delivery.

Mapped to Spring IoC:

Real Life	Spring Framework	Role
The customer	Main class/app	Wants to "use" a service
Zomato App	Spring IoC Container	Takes care of finding, creating, managing
Restaurant	The actual class (e.g., StripePayment)	Where the real work is done
Delivery Boy	Spring Bean	The ready-to-use object Spring gives you

So a better phrasing would be:

"Spring IoC Container is like Zomato. Spring Beans are the food or services being delivered."
You (the app) just place an order, and **Spring** gives you the **ready object (bean).**

Final Breakdown:

- You **don't create beans** manually (you don't cook the food)
- You just **ask Spring** to give you what you need (you place an order)
- Spring (like Zomato) delivers it to you (via dependency injection)

- Zomato = **Spring IoC Container** (manages and delivers services)
- Food = **Service class (e.g., Delivery)**
- Delivery boy = **The Spring Bean carrying the service to your app**

Types of Dependency Injection

There are mainly three types of Dependency Injection:

- **Constructor Injection:** In this type of injection, the injector supplies dependency through the client class constructor.
- **Setter Injection / Property Injection:** In this type of injection, the injector method injects the dependency to the setter method exposed by the client.
- **Interface Injection:** In this type of injection, the injector uses Interface to provide dependency to the client class. The clients must implement an interface that will expose a setter method which accepts the dependency.

Dependency Injection is responsible to create objects, understand which classes require those objects and finally provide those classes with the objects.

Benefits of Dependency Injection

Consider a scenario, wherein you have an Email Class, whose sole responsibility is to take care of the emails received. Now, this class will have objects such as “To email address”, “From email address”, “Subject and the Body of the email”.

Now, if the company wants to save text and audio messages, do you think this class can save the message?

Well, the answer is no?

That’s because, the Email Class cannot handle the parameters of the text and the audio messages. In such cases, you will have to recreate the class. Now, recreating the class is quite a cumbersome job, especially if you have to it regularly.

Instead, if you use Dependency Injection, you can change the objects at run-time. So, in this way, you do not have to recreate the class which further helps you in a lot of ways.