Introduction to Spring Framework

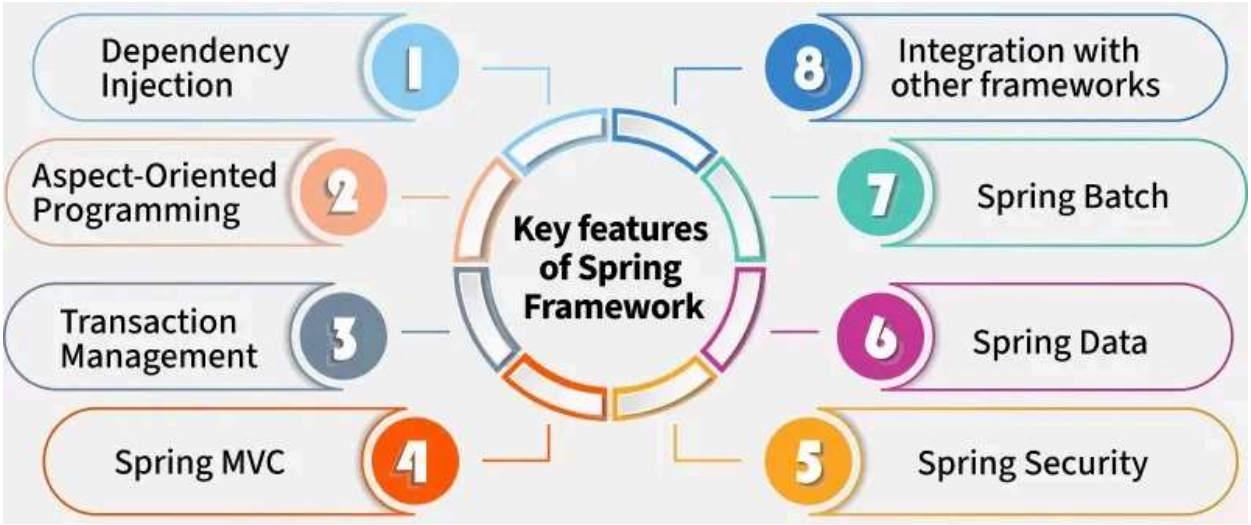
Last Updated : 04 Aug, 2025

The Spring Framework is a lightweight Java framework widely used for building scalable, maintainable enterprise applications. It offers a comprehensive programming and configuration model for Java-based development.

Benefits of Using Spring Framework

- **Simplified Development:** Spring reduces boilerplate code with features like Dependency Injection and AOP, making development faster and easier.
- **Loose Coupling:** Dependency Injection ensures components are loosely coupled, improving maintainability and testability.
- **Modular:** Spring's modular architecture allows developers to use only the required components, improving flexibility and efficiency.
- **Integration Support:** Spring provides built-in support for various technologies like JDBC, JMS and JPA, making integration with other systems seamless.
- **Scalability:** Spring's lightweight nature and support for various web and enterprise components make it highly scalable for large applications.

Key Features of Spring Framework



Key Features of spring framework

- **Dependency Injection:** Dependency Injection is a design pattern where the Spring container automatically provides the required dependencies to a class, instead of the class creating them itself. This promotes loose coupling, easier testing and better maintainability by decoupling the object creation and usage.
- **Aspect-Oriented Programming (AOP):** AOP allows developers to separate cross-cutting concerns (such as logging, security and transaction management) from the business logic.
- **Transaction Management:** Spring provides a consistent abstraction for managing transactions across various databases and message services.
- **Spring MVC:** It is a powerful framework for building web applications that follow the Model-View-Controller pattern.
- **Spring Security:** Spring Security provides comprehensive security features including authentication, authorization and protection against common vulnerabilities.
- **Spring Data:** Spring Data is a part of the Spring Framework that simplifies database access by providing easy-to-use abstractions for working with relational and non-relational databases.
- **Spring Batch:** Spring Batch is a framework in Spring for handling large-scale batch processing, such as reading, processing and writing data in bulk.

- **Integration with Other Frameworks:** Spring integrates seamlessly with other technologies like Hibernate, JPA, JMS and more, making it versatile for various enterprise applications.

Concepts of Spring Framework

1. Dependency Injection

Dependency Injection is a design pattern used in software development to implement Inversion of Control. It allows a class to receive its dependencies from an external source rather than creating them within the class. This reduces the dependency between classes and makes the system more maintainable.

Types of Dependency Injection

Constructor Injection: In constructor injection, the dependent object is provided to the class via its constructor. The dependencies are passed when an instance of the class is created.

Example: This example demonstrates, the Car class depends on the Engine class and the dependency is provided via the constructor.

```
// Constructor Injection
public class Car {
    private Engine engine;

    public Car(Engine engine) {
        this.engine = engine;
    }
}
```

Setter Injection: In setter injection, the dependent object is provided to the class via a setter method after the class is instantiated. This allows you to change the dependencies dynamically.

Example: This example, demonstrates the Car class receives the Engine class through a setter method, which is called after the Car object is created.

```
// Setter Injection
public class Car {
    private Engine engine;

    public void setEngine(Engine engine) {
        this.engine = engine;
    }
}
```

Field Injection: In field injection, the dependent object is directly injected into the class through its fields. It is done using framework like Spring(via annotations). The Dependency Injection automatically injects the dependency without requiring explicit constructor or setter methods.

Example: This example, demonstrates the Engine class is injected directly into the Car class through its field, using annotations or DI frameworks.

```
// Field Injection
public class Car {
    @Autowired
    private Engine engine;
}
```

2. Inversion of Control (IOC) Container

Inversion of Control (IoC) is a design principle used in object-oriented programming where the control of object creation and dependency management is transferred from the application code to an external framework or container. This reduces the complexity of managing dependencies manually and allows for more modular and flexible code.

In Spring framework there are mainly two types of IOC Container which are listed below:

BeanFactory: BeanFactory is the simplest container and is used to create and manage beans. It is a basic container that initializes beans lazily (i.e., only when they are needed). It is typically used for lightweight applications where the overhead of ApplicationContext is not required.

Example:

```
Resource resource = new ClassPathResource("beans.xml");
BeanFactory factory = new XmlBeanFactory(resource);
MyBean obj = (MyBean) factory.getBean("myBean");
```

Explanation:

- ClassPathResource loads the beans.xml file from the classpath.
- XmlBeanFactory creates a basic IoC container using that XML (loads beans lazily).
- getBean("myBean") retrieves and creates the bean with ID myBean.

Note: This will create a Car bean inside the IoC container, which will be initialized when requested.

Application Context: ApplicationContext is an advanced container that extends BeanFactory and provides additional features like internationalization support, event propagation and AOP (Aspect-Oriented Programming) support. The ApplicationContext is preferred in most Spring applications because of its enhanced features.

Example:

```
ApplicationContext context = new
ClassPathXmlApplicationContext("beans.xml");
MyBean obj = (MyBean) context.getBean("myBean");
```

3. Spring Annotation

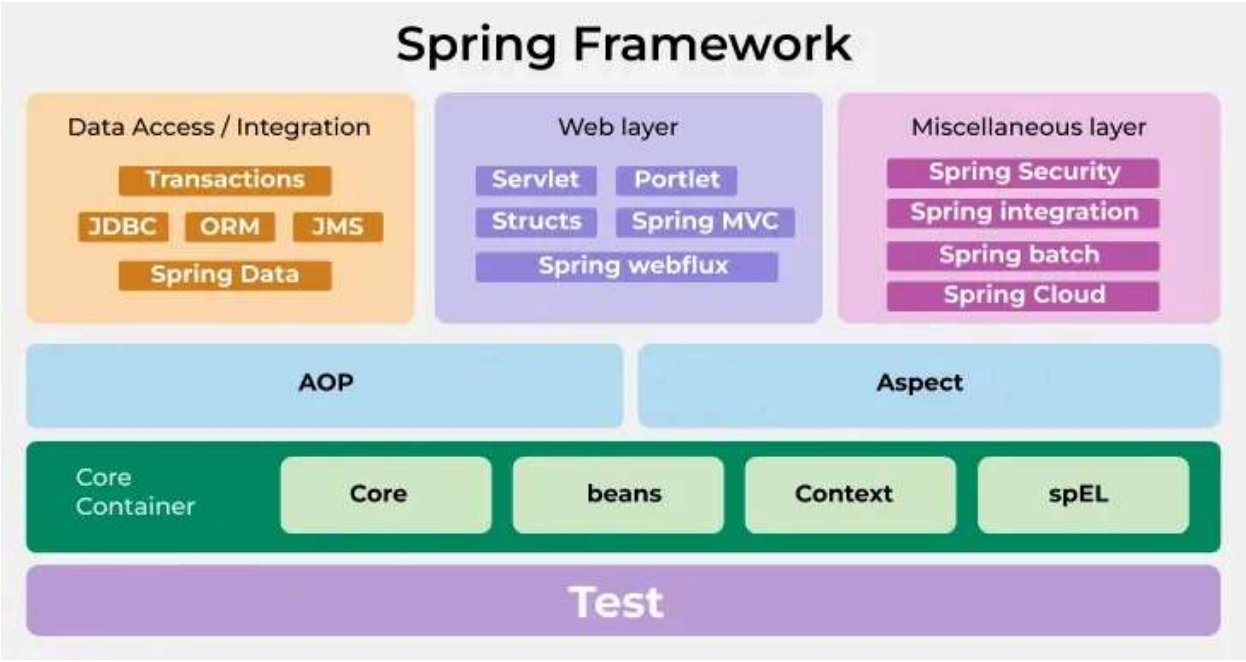
Spring Annotations are metadata used by the Spring Framework to define configuration, dependencies and behavior directly in Java code. They allow Spring to automatically detect, create and manage beans at runtime using component scanning and reflection.

Lets discuss some common type of annotation:

- **@Component:** Marks a class as a Spring bean, allowing Spring to automatically detect and manage it during classpath scanning.
- **@Autowired:** Automatically injects dependencies into a class. It can be used on fields, constructors or methods, allowing Spring to resolve and inject the required beans.
- **@Bean:** Defines a Spring bean explicitly within a configuration class. This is used to create and configure beans that are not automatically detected by classpath scanning.
- **@Configuration:** Indicates that a class contains bean definitions and acts as a source of bean configuration. It is used to mark a class as a configuration class that contains methods annotated with @Bean to define beans.

These annotations are key for managing the Spring IoC (Inversion of Control) container and defining dependencies in our application.

Architecture of Spring Framework



The Spring framework consists of seven modules which are shown in the above Figure. These modules are Spring Core, Spring AOP, Spring Web MVC, Spring DAO, Spring ORM, Spring context and Spring Web flow. These modules provide different platforms to develop different

enterprise applications; for example, you can use Spring Web MVC module for developing MVC-based applications.

Spring Framework Modules

- **Spring Core Module:** The core component providing the IoC container for managing beans and their dependencies. It includes BeanFactory and ApplicationContext for object creation and dependency injection.
- **Spring AOP Module:** Implements Aspect-Oriented Programming to handle cross-cutting concerns like transaction management, logging and monitoring, using aspects defined with the @Aspect annotation.
- **Spring ORM Module:** Provides APIs for database interactions using ORM frameworks like JDO, Hibernate and iBatis. It simplifies transaction management and exception handling with DAO support.
- **Spring Web MVC Module:** Implements the MVC architecture to create web applications. It separates model and view components, routing requests through the DispatcherServlet to controllers and views.
- **Spring DAO Module:** Provides data access support through JDBC, Hibernate or JDO, offering an abstraction layer to simplify database interaction and transaction management.
- **Spring Application Context Module:** Builds on the Core module, offering enhanced features like internationalization, validation, event propagation and resource loading via the ApplicationContext interface.

Spring vs Java EE vs Hibernate

Features	Spring	Java EE	Hibernate
Types	It is a lightweight, modular framework for enterprise applications.	It is a heavyweight platform providing a comprehensive set of services.	It is a framework focuses on ORM(Object Relational Mapping)

Features	Spring	Java EE	Hibernate
Modularity	Highly modular means we can use only the needed components.	It comes with a large predefine set of APIs	It entirely focuses on database interaction
Focus Areas	Focuses on Dependency Injection, AOP and MVC for scalability.	Primarily focuses on enterprise-level services like messaging and transactions.	Simplifies database operations, mapping objects to tables.
Transaction Management	Flexible, supports both declarative and programmatic transactions.	Centralized, relies on JTA for managing transactions.	Manages database transactions, but no broader transaction services.

Evolution of Spring Framework

The framework was first released under the Apache 2.0 license in June 2003. After that there has been a significant major revision, such as Spring 2.0 provided XML namespaces and AspectJ support, Spring 2.5 provide annotation-driven configuration, Spring 3.0 provided a Java-based @Configuration model. As of 2025, the latest release is **Spring Framework 6**, which supports Java 17+, Jakarta EE 10 and native compilation using GraalVM. Earlier major versions include Spring 5 (supporting reactive programming and Java 8+). Though you can still use Spring with an older version of java, the minimum requirement is restricted to Java SE 6. Spring 4.0 also supports Java EE 7 technologies,

such as java message service (JMS) 2.0, java persistence API (JPA) 2.1, Bean validation 1.1, servlet 3.1 and JCache.

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305