Spring Framework

Provides core support for dependency injection, transaction management, web apps, data access, messaging and more.

* The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications – on any kind of deployment platform.
* A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Features:

1. **Core Technologies** **–** dependency injection, events, resources, i18n, validation, data binding,

type Conversion, SpEL, AOP.

1. **Testing –** mock objects, TestContext Framework, Spring MVC Test, WebTestClient.
2. **Data Access** – transactions, DAO Support, JDBC, ORM, Marshalling XML
3. **Spring MVC** and **Spring WebFlux** web frameworks.
4. **Integration –** remoting, JMS, JCA, JMX, email, tasks, scheduling, cache.
5. **Languages –** Kotlin, Groovy, dynamic languages.

---------------------------------------------------------------------------------------------------------------------------------------------------

**Core Technologies**

**================**

* All the technologies are absolutely integral to the Spring Framework.
* Inversion of Control container, Aspect Oriented Programming (AOP).

The IOC Container

--------------------------

* Inversion Of Control (IOC) principle also known as Dependency Injection (DI).
* It is a process whereby objects define their dependencies (that is, the other objects they work with) only through constructor arguments, arguments to a factory method, or properties that are set on the object instance after it is constructed or returned from a factory method.
* The container then injects those dependencies when it creates the bean. This process is fundamentally the inverse (hence the name, Inversion of Control) of the bean itself controlling the instantiation or location of its dependencies by using direct construction of classes or a mechanism such as the Service Locator pattern.
* The org.springframework.beans and org.springframework.context packages are the basis for Spring Framework’s IoC container.
* The [BeanFactory](https://docs.spring.io/spring-framework/docs/5.2.1.RELEASE/javadoc-api/org/springframework/beans/factory/BeanFactory.html) interface provides an advanced configuration mechanism capable of managing any type of object.
* [ApplicationContext](https://docs.spring.io/spring-framework/docs/5.2.1.RELEASE/javadoc-api/org/springframework/context/ApplicationContext.html) is a sub-interface of BeanFactory. It adds:
* Easier integration with Spring’s AOP features
* Message resource handling (for use in internationalization)
* Event publication
* Application-layer specific contexts such as the WebApplicationContext for use in web applications.
* In short, the BeanFactory provides the configuration framework and basic functionality, and the ApplicationContext adds more enterprise-specific functionality.
* In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container.
* The org.springframework.context.ApplicationContext interface represents the **Spring IoC container** and is responsible for instantiating, configuring, and assembling the beans.
* The container gets its instructions on what objects to instantiate, configure, and assemble by reading **configuration metadata**.
* The configuration metadata is represented in XML, Java annotations, or Java code. It lets you express the objects that compose your application and the rich interdependencies between those objects.
* Several implementations of the ApplicationContext interface are supplied with Spring.
* In stand-alone applications, it is common to create an instance of [ClassPathXmlApplicationContext](https://docs.spring.io/spring-framework/docs/5.2.1.RELEASE/javadoc-api/org/springframework/context/support/ClassPathXmlApplicationContext.html) or [FileSystemXmlApplicationContext](https://docs.spring.io/spring-framework/docs/5.2.1.RELEASE/javadoc-api/org/springframework/context/support/FileSystemXmlApplicationContext.html).
* The Spring IoC container consumes a form of configuration metadata.
* This configuration metadata represents how you, as an application developer, tell the Spring container to instantiate, configure, and assemble the objects in your application.
* Configuration metadata can be in the form of XML, Annotation-Based or Java-Based.
* Typically, you define service layer objects, data access objects (DAOs), presentation objects such as Struts Action instances, infrastructure objects such as Hibernate SessionFactories, JMS Queues, and so forth. Typically, one does not configure fine-grained domain objects in the container, because it is usually the responsibility of DAOs and business logic to create and load domain objects.
* The location path or paths supplied to an ApplicationContext constructor are resource strings that let the container load configuration metadata from a variety of external resources, such as the local file system, the Java CLASSPATH, and so on.
* The ApplicationContext is the interface for an advanced factory capable of maintaining a registry of different beans and their dependencies.
* By using the method T getBean(String name, Class<T> requiredType), you can retrieve instances of your beans.
* your application code should have no calls to the getBean() method at all and thus have no dependency on Spring APIs at all.
* Spring’s integration with web frameworks provides dependency injection for various web framework components such as controllers and JSF-managed beans, letting you declare a dependency on a specific bean through metadata (such as an autowiring annotation).
* A Spring IoC container manages one or more beans. These beans are created with the configuration metadata that you supply to the container (for example, in the form of XML <bean/> definitions).