**Spring Framework Architecture**

The basic idea behind the development of Spring Framework was to make it a one-stop shop where you can integrate and use modules according to the need of your application. This modularity is due to the architecture of [Spring](https://data-flair.training/blogs/spring-framework-tutorial/). There are about 20 modules in the Spring framework that are being used according to the nature of the application.

Below is the architecture diagram of the Spring framework. There, you can see all the modules that are being defined over the top of Core container. This layered architecture contains all the necessary modules that a developer may require in developing an enterprise application. Also, the developer is free to choose or discard any the  module that is of need or are not of any use according to its requirement. Due to its modular architecture, the integration of [Spring framework](https://data-flair.training/blogs/spring-framework-environment-setup/) with other frameworks is super easy.

**Modules of Spring Architecture**

The Spring framework architecture contains four major modules. Let's get into it!

**Core Container**

The Core Container in the Spring architecture contains the Core, Beans, Context, and Expression Language.

* The ***Core***provides features to fundamental parts of the framework. This includes the *Ioc* as well the *Dependency Injection* (DI).
* A sophisticated implementation of the factory pattern, called BeanFactory*,*is provided by the **Bean** module.
* The ***Context*** module uses the base provided by the Core and Beans modules. It has access rights for any objects that define and configure. In the Context module, the major point is the ApplicationContext interface.
* In the Spring framework, a powerful language tool for querying is given by the ***Expression Language Module***. It also provides manipulation of an object graph at run-time execution.

**Data Access/Integration**

In this part of the Spring architecture, we are going to discuss OXM, JMS, ORM, Transaction Modules, and JDBC.

* The***JDBC*** module contains a JDBC-abstraction layer, which removes the need for JDBC related coding.
* The Spring framework doesn’t have its own ***ORM***implementation. But, it offers the integration layers with other popular Object-Relational Mapping tools, such as iBATIS, Hibernate, etc.
* Spring ***OXM*** (Spring Object XML Mappers), a Spring framework module. With the help of this module, we can ease the mappings between Java objects and XML documents. The module can extend. Hence, it provides integration with various other popular frameworks such as JAXB, Castor, XStream, etc.
* The***JMS*** module, or Java Messaging Services, is a Spring module used for sending and receiving messages.
* The***Transaction***Management module is for coordinating Java objects' transaction. It is also used to unify several transaction management APIs.

**The Web Container**

The Web module in[Java](https://data-flair.training/blogs/java-tutorial/) Spring contains several frameworks, used to ease the task of developing web-related applications. Various popular MVC frameworks, like JSF, Struts, etc., can integrate with the Web module. The Web layer has various modules, such as Web, Web-MVC, Web-Socket, and Web-Portlet, which are as follows:

* The ***Web*** module has features of web-oriented integrations. For example, web-oriented application context, multiple file upload functions, and initialization of IoC using a servlet.
* The Spring ***Web-MVC*** module is a Model-View-Controller (MVC) of Spring. It provides the implementation for web applications.
* The***Web-Socket*** module of the Spring Framework provides support for WebSocket. It provides bi-direction communication between the server side and client side in web-related applications.
* The ***Web-Portlet***Spring module insists on implementing MVC in a portlet environment. The functionality of Web-Portlet is same as that of the Web-Socket module of Spring.

**Miscellaneous**

There are several important modules along with AOP, Aspects of Instrumentation, Messaging, and Test.

* The ***AOP***, *Aspect Oriented Programming,* is a Spring framework module. It implements cross-cutting concerns. It can alternatively understand an interceptor that intercepts some processes, like a method that is intercepted by Spring AOP during its execution for adding an extra functionality.
* The***Aspects***Spring module is a mature and powerful AOP framework. Using it, Spring AspectJ integration is possible.
* The ***Testing*** module is for providing support to classes for writing integration and unit tests. It uses JUnit or TestNG type frameworks.
* Testing the ***Instrumentation*** module in Spring Framework is for providing class instrument support. These modules are used in various application servers.
* The ***Messaging*** module composes configurative registration of message objects for message consumption from message queues.

