Spring Framework

Spring is a *lightweight* framework. It can be thought of as a *framework of frameworks* because it provides support to various frameworks such as [Struts](https://www.javatpoint.com/struts-2-tutorial), [Hibernate](https://www.javatpoint.com/hibernate-tutorial), [EJB](https://www.javatpoint.com/ejb-tutorial), [JSF](https://www.javatpoint.com/jsf-tutorial), etc.

### **Inversion Of Control (IOC) and Dependency Injection**

These are the design patterns that are used to remove dependency from the programming code. They make the code easier to test and maintain. Let's understand this with the following code:

1. **class** Employee{
2. Address address;
3. Employee(){
4. address=**new** Address();
5. }
6. }

In such case, there is dependency between the Employee and Address (tight coupling). In the Inversion of Control scenario, we do this something like this:

1. **class** Employee{
2. Address address;
3. Employee(Address address){
4. **this**.address=address;
5. }
6. }

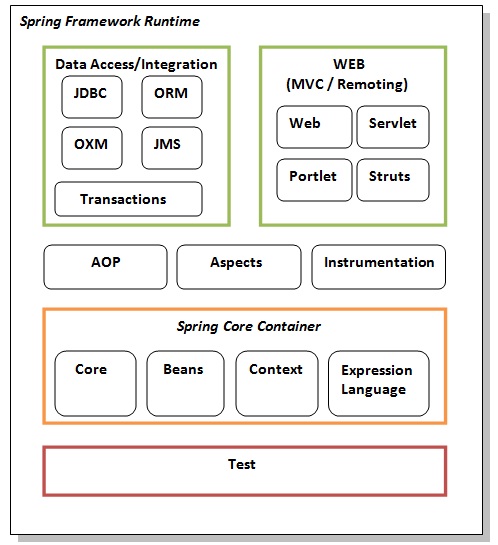
Thus, IOC makes the code loosely coupled. In such case, there is no need to modify the code if our logic is moved to new environment.

In Spring framework, IOC container is responsible to inject the dependency. We provide metadata to the IOC container either by XML file or annotation.

#### **Advantage of Dependency Injection**

* makes the code loosely coupled so easy to maintain
* makes the code easy to test

#### **Spring Architecture**



ORM – Object Relational Mapping

Spring OXM – Spring Object XML Mappers

JMS – Java Messaging Service

WEB – WEB MVC

Portlet – Loading by segment by segment, No direct URL Call, No form data transfer, Similar to servlet in many aspects

Struts – Supports servlet, JSP, TAG Library, etc

AOP – Supports aspect in spring bean

Aspect – supports aspectj which in turn supports Aspects anywhere

Instrumentation – provides class loader

Core and Beans – IoC and DI

Context – Resource Injection,  internationalization (I18N), EJB, JMS, Basic Remoting.

Expression Language - support to setting and getting property values, method invocation, accessing collections and indexers, named variables, logical and arithmetic operators, retrieval of objects by name etc.

**Example**

**Student.java**

**public** **class** Student {

**private** String name;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** displayInfo(){

    System.out.println("Hello: "+name);

}

}

**applicationContext.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans

    xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xmlns:p="http://www.springframework.org/schema/p"

    xsi:schemaLocation="http://www.springframework.org/schema/beans

               http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="studentbean" **class**="com.javatpoint.Student">

<property name="name" value="Vimal Jaiswal"></property>

</bean>

</beans>

**Test.java**

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.ClassPathResource;

**import** org.springframework.core.io.Resource;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

    Resource resource=**new** ClassPathResource("applicationContext.xml");

    BeanFactory factory=**new** XmlBeanFactory(resource);

    Student student=(Student)factory.getBean("studentbean");

    student.displayInfo();

}

}