**INTERNITY FOUNDATION**

**TASK-14**

**Submitted By:**

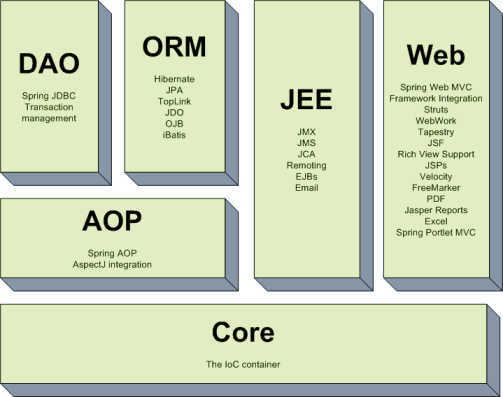
**Amisha Singhal**

**Java Batch**

**Spring - Introduction and Architecture**

**Ans-** [Spring Framework](https://www.upgrad.com/blog/spring-interview-questions-answers/) is a Java platform from the Java enterprise release which presents an extensive infrastructure practice for developing Java-based applications. Using the spring’s infrastructure technique, a developer can easily build the application from the **plain old Java objects**. This skill applies to the[Java](https://www.upgrad.com/blog/how-to-code-compile-and-run-java-projects/) SE [programming](https://www.upgrad.com/blog/top-programming-languages-to-learn/) standard with full and partial Java EE (Enterprise Edition) techniques.

The Spring architecture has:



**The core module:**provides the dependency injection (di) feature which is the basic concept of the spring framework. this module contains the ***beanfactory,***an implementation of factory pattern which creates the bean as per the configurations provided by the developer in an xml file.

**AOP module:**the aspect oriented programming module allows developers to define method-interceptors and point cuts to keep the concerns apart. it is configured at run time so the compilation step is skipped. it aims at declarative transaction management which is easier to maintain.

**DAO module:**this provides an abstraction layer to the low level task of creating a connection, releasing it etc. it also maintains a hierarchy of meaningful exceptions rather than throwing complicated error codes from specific database vendors. it uses aop to manage transactions. transactions can also be managed programmatically.

**Orm module:**spring doesn’t provides its own orm implementation but offers integrations with popular object relational mapping tools like hibernate, ibatis sql maps, oracle toplink and jpa etc.

**Jee module:**it also provides support for jmx, jca, ejb and jms etc. in lots of cases, jca (java ee connection api) is much like jdbc, except where jdbc is focused on database jca focus on connecting to legacy systems.

**Web module:**spring comes with mvc framework which eases the task of developing web applications. it also integrates well with the most popular mvc frameworks like struts, tapestry, jsf, wicket etc.

**Spring - IoC Containers**

**Ans-**The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

* to instantiate the application class
* to configure the object
* to assemble the dependencies between the objects

**For instance:** Factory, Abstract Factory, Builder,Decorator to create and develop the numerous classes and object instances for creating the extensive Java applications.

The Spring Framework Inversion of Control (IoC) provides the best practice using the design patterns as first-class objects that we can easily use in applications. Many organisations use the Spring Framework for creating robust and scalable applications.

There are two types of IoC containers. They are:

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

**Spring - Bean Definition**

**Ans-** Spring Beans are the objects that form the backbone of the application. These are managed by the [**Spring IoC container**](https://data-flair.training/blogs/spring-ioc-containers/). It does it by instantiating and assembling the bean object. These objects are created with the configuration metadata which is applied to the container in the form of XML <bean/> .  
The configuration metadata which contains the information about Spring Bean definitions needs the container to know the things like:

* Creating the Bean.
* Life-cycle details of Bean.
* Spring Bean Dependencies.

Every object that is under the control of Spring’s ApplicationContext in terms of creation, orchestration, and destruction is called a Spring Bean.

**Spring - Bean Scopes**

**Ans-** There are five types of [spring bean](https://www.journaldev.com/2461/spring-ioc-bean-example-tutorial) scopes:

1. **singleton** – only one instance of the spring bean will be created for the spring container. This is the default spring bean scope. While using this scope, make sure bean doesn’t have shared instance variables otherwise it might lead to data inconsistency issues.

**Example of singleton scope bean using XML config –**

|  |
| --- |
| **<!-- To specify singleton scope is redundant -->**  **<bean id="beanId" class="com.howtodoinjava.BeanClass" scope="singleton" />**  **//or**  **<bean id="beanId" class="com.howtodoinjava.BeanClass" />** |

1. **prototype** – A new instance will be created every time the bean is requested from the spring container.

**XML config example of prototype bean scope –**

|  |
| --- |
| **<bean id="beanId" class="com.howtodoinjava.BeanClass" scope="prototype" />** |

1. **request** – This is same as prototype scope, however it’s meant to be used for web applications. A new instance of the bean will be created for each HTTP request.

**XML config example of request bean scope –**

|  |
| --- |
| **<bean id="beanId" class="com.howtodoinjava.BeanClass" scope="request" />** |

1. **session** – A new bean will be created for each HTTP session by the container.

**XML config example of session bean scope –**

|  |
| --- |
| **<bean id="beanId" class="com.howtodoinjava.BeanClass" scope="session" />** |

1. **global-session** – This is used to create global session beans for Portlet applications.

**XML config example of application bean scope –**

|  |
| --- |
| **<bean id="beanId" class="com.howtodoinjava.BeanClass" scope="application" />** |

**Spring - Bean Life Cycle**

**Ans-** When container starts,a Spring bean needs to be instantiated, based on Java or XML bean definition. It may also be required to perform some post-initialization steps to get it into a usable state. After that, when the bean is no longer required, it will be removed from the IoC container.

Spring bean factory is responsible for managing the life cycle of beans created through spring container.

#### 1. Life cycle callbacks

Spring bean factory controls the creation and destruction of beans. To execute some custom code, it provides the call back methods which can be categorized broadly in two groups:

* **Post-initialization** call back methods
* **Pre-destruction** call back methods

## 2. Life cycle callback methods

Spring framework provides following **4 ways for controlling life cycle events** of a bean:

1. InitializingBean and DisposableBean callback interfaces
2. \*Aware interfaces for specific behavior
3. Custom init() and destroy() methods in bean configuration file
4. @PostConstruct and @PreDestroy annotations

