**Spring**

1] **Spring framework :-**

Spring is light weight ,open source framework meant for reducing the complexity of developing enterprise-level- application. It is also called “Framework of Frameworks”, as spring provides support to various other important framework.

Spring is a Dependency Injection framework to make java Application ‘Loosely Coupled”.

-Loose coupling allows us making changes in the code easily.

- Testing of loosely coupled structures is easier then tightly coupled structures.

Exa ;

public class A extends B {

Demo i = new B ();

}

public class B implements Demo {

}

public interface demo {

}

Note –

class A is compatible with any class . In future , if we don’t want B class then we can create the one more class named as class C and pass the object.

Demo i = new C () ;

Here, i is the interface reference.

**What is Spring Boot ? Why did you use Spring boot in your project ? Why not Spring ?**

Spring Boot is a Spring Module. Spring boot is a framework for Rapid Application Development build using Spring framework with extra support of Auto-Configuration and embedded application Server like Tomcat, Jetty .

Why :- Because it provides “Rapid Application Development”. Spring requires much of the boiler plate codes for configurations and dependency injections, all those things removed in spring boot. Hence , we can use spring boot rather than just spring framework.

**What is the use of profiles in Spring Boot ?**

- Suppose we create a Spring boot Application , that time we deal with different environments (like Development, Testing, production).

In development environment – we use H2 database & in production environment we use Oracle/ MySQL. In this case Setting will be change.

So for that , Spring has the provision of profiles to keep the separate configuration of each Environments.

**What are the features of Spring Framework?**

· Spring framework follows layered architecture pattern that helps in the necessary components selection along with providing a robust and cohesive framework for J2EE applications development.

· The AOP (Aspect Oriented Programming) part of Spring supports unified development by ensuring separation of application’s business logic from other system services.

· Spring provides highly configurable MVC web application framework which has the ability to switch to other frameworks easily.

· Provides provision of creation and management of the configurations and defining the lifecycle of application objects.

· Spring has a special design principle which is known as IoC (Inversion of Control) that supports objects to give their dependencies rather than looking for creating dependent objects.

· Spring is a lightweight, java based, loosely coupled framework.

· Spring provides generic abstraction layer for transaction management that is also very useful for container-less environments.

· Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate or other frameworks) into consistent, unchecked exceptions. This introduces abstraction and greatly simplifies exception handling.

What is Spring Framework? · Spring is a powerful open-source, loosely coupled, lightweight, java framework meant for reducing the complexity of developing enterprise-level applications. This framework is also called the “framework of frameworks” as spring provides support to various other important frameworks like JSF, Hibernate, Structs, EJB, etc.

· There are around 20 modules which are generalized into the following types:

o Core Container

o Data Access/Integration

o Web

o AOP (Aspect Oriented Programming)

o Instrumentation

o Messaging

o Test

**What is a Spring configuration file?**

A Spring configuration file is basically an XML file that mainly contains the classes information and describes how those classes are configured and linked to each other. The XML configuration files are verbose and cleaner.

**What do you mean by IoC (Inversion of Control) Container?**

Spring container forms the core of the Spring Framework. The Spring container uses Dependency Injection (DI) for managing the application components by creating objects, wiring them together along with configuring and managing their overall life cycles. The instructions for the spring container to do the tasks can be provided either by XML configuration, Java annotations, or Java code.

What do you understand by Dependency Injection?

-The main idea in Dependency Injection is that you don’t have to create your objects but you just have to describe how they should be created.

· The components and services need not be connected by us in the code directly. We have to describe which services are needed by which components in the configuration file. The IoC container present in Spring will wire them up together.

In Java, the 2 major ways of achieving dependency injection are:

o Constructor injection: Here, the IoC container invokes the class constructor with a number of arguments where each argument represents a dependency on the other class.

o Setter injection: Here, the spring container calls the setter methods on the beans after invoking a no-argument static factory method or default constructor to instantiate the bean.

Explain the difference between constructor and setter injection?

· In constructor injection, partial injection is not allowed whereas it is allowed in setter injection.

· The constructor injection doesn’t override the setter property whereas the same is not true for setter injection.

· Constructor injection creates a new instance if any modification is done. The creation of a new instance is not possible in setter injection.

· In case the bean has many properties, then constructor injection is preferred. If it has few properties, then setter injection is preferred.

· **What are Spring Beans?**

· They are the objects forming the backbone of the user’s application and are managed by the Spring IoC container.

· Spring beans are instantiated, configured, wired, and managed by IoC container.

· Beans are created with the configuration metadata that the users supply to the container (by means of XML or java annotations configurations.)

·

· **What are the bean scopes available in Spring?**

The Spring Framework has five scope supports. They are:

· Singleton: The scope of bean definition while using this would be a single instance per IoC container.

· Prototype: Here, the scope for a single bean definition can be any number of object instances.

· Request: The scope of the bean definition is an HTTP request.

· Session: Here, the scope of the bean definition is HTTP-session.

· Global-session: The scope of the bean definition here is a Global HTTP session.

**What is autowiring and name the different modes of it?**

The IoC container autowires relationships between the application beans. Spring lets collaborators resolve which bean has to be wired automatically by inspecting the contents of the BeanFactory. Different modes of this process are:

· no: This means no autowiring and is the default setting. An explicit bean reference should be used for wiring.

· byName: The bean dependency is injected according to the name of the bean. This matches and wires its properties with the beans defined by the same names as per the configuration.

· byType: This injects the bean dependency based on type.

· constructor: Here, it injects the bean dependency by calling the constructor of the class. It has a large number of parameters.

· autodetect: First the container tries to wire using autowire by the constructor, if it isn't possible then it tries to autowire by byType.

What are the limitations of autowiring?

· Overriding possibility: Dependencies are specified using <constructor-arg> and <property> settings that override autowiring.

· Data types restriction: Primitive data types, Strings, and Classes can’t be autowired