1. **What is a spring framework?**

It is an open source java application framework, which supports building of all types of java applications like web application, DB driven application, Enterprise application

**Note:** What is the diff between web application and Enterprise application?

1. **What are the features of spring application?**

Note: **Framework:** collection of libraries and jars, Gives basic configuration to build the application

**Lightweight:** Spring jars are smaller (that is < 10 MB SIZE) and these do not need heavy weight application servers to.

**Non-Invasive:** No need to extend/implement any predefined classes

**Example:** For normal jdbc connection, we have to write so much of code.

**Loosely Coupled:**

Dependency injection, AOP helps in creating loosely coupled code base.

**IOC:** lt allows creation of objects that will be managed by spring container and its dependency too will be resolved automatically.

**Spring Container:** It takes care of object creation, initialization and managing object dependencies.

**AOP:** promotes separation of concerns (Supporting functions) such as logging, transaction security from core business logic.

1. **What are the different modules in spring framework?**

**Core Container:** This module provides IOC and Dependency Injection features to spring framework. It also contains Bean factory, application context and spEL (spring expression language)

**Data Access/Integration:**

Provides support to interact with db and to integrate with ORM solution like Hibernate, JPA.

**Web:**

This Web module provides the basic support for web application development. It has web application context, which is built on application context of core container. Hence, it provides complete MVC implementation to develop presentation tier and supports integration with other frameworks like JSF, struts etc.

**Others:**

Other modules also provided like aop, messaging, testing like junits etc.

****

1. **What do you mean by IOC (Inversion of Control) Container?**

Represents the inversion of responsibility of object creation, initialization and destruction (life cycle management) from application to spring container.

In normal programming, we create object with New Keyword

In spring, we depend on container to provide us with the required dependency object.

1. **Explain the different ways to configure a Java class as Spring Bean?**

**Xml and Java Way:**

**Performance and Readability is more:**

All Bean definitintions, scope definitions, dependency mechanism defined in a single file, so it will improve the performance and readability. No needs to scan any packages for bean creation and there is component scanning.

Redundancy:As all definitions are placed in a single file, there might be chance to duplicate e bean in spring.

**Annotation based:**

**Time Reduction**

**Redundancy is low.**

**Performance is low:** If we have more than one package, spring container scan all the packages to know which lasses are eligible for bean creation.

1. **What are the different types of containers in spring and difference?**

**Key Bean Factory Application context**

**Implementation** XML bean factory implements FileSystemXmlApplication

Bean factory Context, ClassPathXmlApp

Context and Annotation config Web Application context implements ApplicationConext.

In addition, Application Context extends Bean factory.

**Annotation Based DI** No Yes

**Instantiation**  Bean factory instantiate (Memory) Application instantiate bean

Beans when get Bean () at the time container start

Method called

**Internationalization** Does not support supports

**Loading Mechanism** Lazy LoadingAggressive Loading

**Spring Aop:**

The most common usage is where your application has cross cutting concerns i.e. a piece of logic or code that is going to be written in multiple classes/layers.

In addition, this could vary based on your needs. Some very common examples of these could be:

1. Transaction Management
2. Logging
3. Exception Handling (especially when you may want to have detailed traces or have some plan of recovering from exceptions)
4. Security aspects
5. Instrumentation
6. Performance metrics

**Terminology:**

Below terminology defined at compile time.

**Aspect:** Aspect is the concern that we are trying to implement generically.

**Pointcut:** Defines an expression, what are the methods that needs to be intercepted.

**Advice:** Exact piece of code what happens when a particular point cut met.

Dynamic – RunTime

**Joint point:** joint point contains all the information like method name, arguments