Q1.What is Spring Framework?

ans:- The Spring Framework (Spring) is an open-source application framework that provides infrastructure support for developing Java applications.

One of the most popular Java Enterprise Edition (Java EE) frameworks, Spring helps developers create high performing applications using plain old Java objects (POJOs).

Spring Framework is a method for product development, when the requirement comes with criteria like the need for dependency Injection (DI), lightweight structure for the product,

with POJO or plain old java object, AOP or aspect oriented programming, an option for unit testing, modular development options, etc. There are many features in Spring Framework that makes it stand out from other product development framework,

such as Spring web services, Spring test content framework, JDBC abstraction layer, Spring MVC framework, a configuration management bundle called IoC container, transaction management, etc.

Q2.What are the features of Spring Framework?

ans:-The Spring Framework contains the following important features:

Inversion of Control (IoC)

The Spring Framework implements the principle of IoC, where the control of object creation and dependency injection is shifted from the application code to the framework.

In traditional programming, objects are typically created and managed by the application code itself. However, with IoC, this responsibility is delegated to a container or framework like Spring.

Here are the benefits of IoC in Spring applications.

Loose coupling: Objects are not tightly coupled to their dependencies, making them easier to replace or modify without affecting other parts of the application.

Testability: Objects can be easily tested in isolation by providing mock or stub dependencies during testing, allowing for comprehensive unit testing.

Reusability: Components can be reused in different contexts or applications, as they are not tied to specific implementations of their dependencies.

Modular design: With IoC, the application code becomes more modular and focused on business logic, while the framework takes care of object creation and management.

Dependency Injection (DI)

Spring supports DI, allowing objects to be injected with their dependencies rather than having to create or manage them explicitly. This promotes easier configuration, better decoupling, and improved testability.

In Spring, DI is achieved through the Spring IoC container. The container is responsible for creating and managing objects, known as beans, and wiring them together by injecting their dependencies. The dependencies are typically declared as interfaces or abstract classes, and the Spring container resolves and provides the concrete implementations at runtime.

There are different ways to perform DI in Spring:

Constructor Injection: Dependencies are injected through a constructor when creating an object. This ensures that the object is fully initialized with its dependencies at the time of creation.

Setter Injection: Dependencies are injected using setter methods. The container calls the setters to provide the necessary dependencies after creating the object.

Field Injection: Dependencies are injected directly into the object's fields using annotations. This approach requires the use of reflection and is less commonly used than constructor or setter injection.

Aspect-Oriented Programming (AOP)

Spring provides AOP capabilities, allowing the modularization of cross-cutting concerns. AOP enables the separation of concerns by providing a way to apply behaviors (aspects) to multiple objects in a declarative manner.

Spring MVC

The Spring MVC (Model-View-Controller) framework provides a robust and flexible architecture for building web applications. It offers features like request mapping, view resolution, data binding, and validation, making it a popular choice for web development.

Transaction Management

Spring offers a comprehensive transaction management abstraction that simplifies handling database transactions. It supports both programmatic and declarative transaction management, with support for various transaction APIs and transactional annotations.

Spring Data

Spring Data provides a consistent and simplified data access framework, integrating with different data storage technologies such as relational databases, NoSQL databases, and more. It offers a unified API and reduces boilerplate code required for common data access operations.

Spring Security

Spring Security is a powerful authentication and authorization framework that helps secure applications. It provides a flexible and customizable approach to handle authentication, authorization, and protection against common security threats.

Spring Boot

Spring Boot is an opinionated framework that simplifies the setup and configuration of Spring applications. It promotes convention over configuration and offers auto-configuration, embedded servers, production-ready features, and seamless integration with other Spring projects.

Testing Support

The Spring Framework provides robust testing support, including support for unit testing, integration testing, and mocking. It offers integration with popular testing frameworks and provides utilities for testing Spring components and applications.

Internationalization and Localization

Spring offers comprehensive support for internationalization and localization, allowing applications to be easily adapted to different languages, regions, and cultures. It provides mechanisms for message resolution, locale management, and resource handling.

These are some of the prominent features of the Spring Framework that contribute to its popularity and make it a versatile and powerful framework for Java application development.

Q3.What is a Spring configuration file?

ans:- A Spring configuration file is an XML file that contains the classes information. It describes how those classes are configured as well as introduced to each other. The XML configuration files, however, are verbose and cleaner.

Q4.What do you mean by IoC Container?

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

There are two types of IoC containers. They are:

BeanFactory

ApplicationContext

Q5.What do you understand by Dependency Injection?

ans:- Dependency Injection (DI) is a design pattern used to implement IoC. It allows the creation of dependent objects outside of a class and provides those objects to a class through different ways. Using DI, we move the creation and binding of the dependent objects outside of the class that depends on them.

The Dependency Injection pattern involves 3 types of classes.

Client Class: The client class (dependent class) is a class which depends on the service class

Service Class: The service class (dependency) is a class that provides service to the client class.

Injector Class: The injector class injects the service class object into the client class.

Types of Dependency Injection

As you have seen above, the injector class injects the service (dependency) to the client (dependent). The injector class injects dependencies broadly in three ways: through a constructor, through a property, or through a method.

Constructor Injection: In the constructor injection, the injector supplies the service (dependency) through the client class constructor.

Property Injection: In the property injection (aka the Setter Injection), the injector supplies the dependency through a public property of the client class.

Method Injection: In this type of injection, the client class implements an interface which declares the method(s) to supply the dependency and the injector uses this interface to supply the dependency to the client class.

Q6.Explain the difference between constructor and setter injection?

ans:- **Setter**

Dependencies are injected using setter methods.

Partial dependency injection is supported; we can ignore any property for injection. We can inject properties only if necessary.

If we use constructor and setter injection both, setter injection will be used because setter methods will be executed after constructor.

Default constructor is mandatory.

More readable

Less secure than constructor inject, because while using setter injection, you can override certain dependency by sub-classes overridden setter methods.

If two objects dependent each other, the circular dependency has no effect as objects use setter methods of other.

Constructor

Dependencies are injected using constructor arguments.

Partial dependency injection is not supported; we cannot ignore the property if it is there in constructor parameter. All the parameters of constructor must be injected. Otherwise an overloaded constructor must be created with those properties need to be injected.

If we use constructor and setter injection, constructor injection will not be used.

Default constructor is not mandatory.

Less Readable when there is more number of properties in class.

More secure than setter inject as dependency cannot be overridden, sub-classes still has to invoke super class constructors.

If two objects dependent each other, the circular dependency has an effect and exception will be thrown from spring. One object has to wait for other object creation; both the objects wait for other creation results in ObjectCurrentlyInCreationException.

Q7.What are Spring Beans?

ans:- Spring beans are the objects which are created and managed completely by spring container.

These beans are the heart of the application.

Beans can be defined in spring either by using XML configuration or by using Annotation.

In XML configuration, bean can be defined using < bean > tag inside < beans > tag.

In Annotation configuration , bean can be defined using the annotations like @Component,@Service,@Controller,@Repository on top of the class definition.

Q8.What are the bean scopes available in Spring?

ans:- There are five types of spring bean scopes:

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.

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

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.

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

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

Q9.**What is Autowiring and name the different modes of it?**

ans:- Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection.

Autowiring can't be used to inject primitive and string values. It works with reference only.

Advantage of Autowiring

It requires the less code because we don't need to write the code to inject the dependency explicitly.

Disadvantage of Autowiring

No control of programmer.

There are many autowiring modes:

No.	Mode	Description
1)	no	It is the default autowiring mode. It means no autowiring bydefault.
2)	byName	The byName mode injects the object dependency according to name of the bean. In such case, property name and bean name must be same. It internally calls setter method.
3)	byTypeThe byType mode injects the object dependency according to type. So property name and bean name can be different. It internally calls setter method.	
4)	constructor	The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having large number of parameters.
5)	autodetect	It is deprecated since Spring 3.

Q10.Explain Bean life cycle in Spring Bean Factory Container.

ans:- Bean life cycle in Spring Bean Factory Container is as follows:

The Spring container instantiates the bean from the bean's definition in the XML file.

Spring populates all of the properties using the dependency injection, as specified in the bean definition.

The factory calls setBeanName() by passing the bean's ID if the bean implements the BeanNameAware interface.

The factory calls setBeanFactory() by passing an instance of itself if the bean implements the BeanFactoryAware interface.

preProcessBeforeInitialization() methods are called if there are any BeanPostProcessors associated with the bean.

If an init-method is specified for the bean, then it will be called.

Finally, postProcessAfterInitialization() methods will be called if there are any BeanPostProcessors associated with the bean.