PPT_JAVA_Assignment-9

Q1.What is Spring Framework?

Spring is a lightweight and popular open-source Java-based framework developed by Rod Johnson in 2003. It is used to develop enterprise-level applications. It provides support to many other frameworks such as Hibernate, Tapestry, EJB, JSF, Struts, etc, so it is also called a framework of frameworks. It's an application framework and IOC (Inversion of Control) container for the Java platform. The spring contains several modules like IOC, AOP, DAO, Context, WEB MVC, etc

Q2.What are the features of Spring Framework?

Features of Spring Framework

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.

Q3.What is a Spring configuration file?

Spring bean configuration file contains spring bean configurations, dependent value configurations, and other miscellaneous configurations. Any name can be given to Spring Bean configuration file with .xml extension. <beans> tag is the root element., this encloses all the spring definitions. <bean> tag defines spring bean i.e. a java class to be initialized and managed by spring core container. Every spring bean class must be configured in spring configuration file, and then only spring container recognizes that class. Every Spring Bean will be identified through its Bean id, which is a value given in id attribute of <bean>.

Q4.What do you mean by IoC Container?

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

Q5.What do you understand by Dependency Injection?

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled.

The Dependency Injection is a design pattern that removes the dependency of the programs. In such case we provide the information from the external source such as XML file. It makes our code loosely coupled and easier for testing.

Q6.Explain the difference between constructor and setter injection?

There are many key differences between constructor injection and setter injection.

1. **Partial dependency**: can be injected using setter injection but it is not possible by constructor. Suppose there are 3 properties in a class, having 3 arg constructor and setters methods. In such case, if you want to pass information for only one property, it is possible by setter method only.
2. **Overriding**: Setter injection overrides the constructor injection. If we use both constructor and setter injection, IOC container will use the setter injection.
3. **Changes**: We can easily change the value by setter injection. It doesn't create a new bean instance always like constructor. So setter injection is flexible than constructor injection

Q7.What are Spring Beans?

A Java class whose object is created and managed by the Spring Container is called a Spring Bean. Moreover, we can consider any class as a Spring Bean, except an abstract class and an interface. It can also be a POJO class, Java Bean class or any other class as well.

Q8.What are the bean scopes available in Spring?

There are five types of spring bean 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.
2. **prototype** – A new instance will be created every time the bean is requested from the spring container.
3. **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.
4. **session** – A new bean will be created for each HTTP session by the container.
5. **global-session** – This is used to create global session beans for Portlet applications.

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

Autowiring Modes

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)	byType	The 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.

Bean life cycle is managed by the spring container. When we run the program then, first of all, the spring container gets started. After that, the container creates the instance of a bean as per the request, and then dependencies are injected. And finally, the bean is destroyed when the spring container is closed. Therefore, if we want to execute some code on the bean instantiation and just after closing the spring container, then we can write that code inside the custom init() method and the destroy() method.