Assignment-9 (Java)

Q1.What is Spring Framework?

Answer: The Spring Framework is a popular open-source Java framework that provides comprehensive infrastructure support for developing enterprise-level Java applications. It offers a modular and layered architecture that promotes loose coupling and easy integration with other frameworks. Spring provides features such as dependency injection, aspect-oriented programming, transaction management, and MVC web framework. It simplifies development tasks and promotes good software engineering practices, making it widely used for building scalable, robust, and maintainable Java applications.

Q2.What are the features of Spring Framework?

Answer: The Spring Framework offers a wide range of features, including:

1. Dependency Injection: Enables loose coupling and promotes modular development.
2. Aspect-Oriented Programming (AOP): Supports cross-cutting concerns, such as logging and transaction management.
3. Spring MVC: Provides a powerful model-view-controller framework for building web applications.
4. Data Access: Offers seamless integration with various data access technologies, including JDBC, JPA, and Hibernate.
5. Transaction Management: Supports declarative transaction management across multiple resources.
6. Security: Provides robust security features like authentication and authorization.
7. Testing: Offers extensive support for testing Spring-based applications with mock objects and integration testing tools.

Q3.What is a Spring configuration file?

Answer: A Spring configuration file, often referred to as an application context file, is an XML or Java-based file that contains configuration information for the Spring Framework. It defines the beans (components) in the application, their dependencies, and other related settings. The configuration file specifies the wiring of components through dependency injection, AOP aspects, and other Spring-specific settings. It serves as a blueprint for the Spring container to create and manage the application's objects, allowing for flexible and modular configuration of the application's components.

Q4.What do you mean by IoC Container?

Answer: IoC (Inversion of Control) Container, also known as the Spring container, is a core feature of the Spring Framework. It manages the lifecycle of objects and controls their dependencies. Instead of objects creating and managing their dependencies, the IoC container takes charge of creating, configuring, and injecting the dependencies into the objects. This inversion of control helps achieve loose coupling, modularity, and easier testing. The container uses configuration metadata, such as XML or annotations, to determine object relationships and manage their lifecycle, promoting a more flexible and modular application design.

Q5.What do you understand by Dependency Injection?

Answer: Dependency Injection (DI) is a design pattern and a core concept in the Spring Framework. It involves injecting the dependencies of an object from an external source, typically managed by

an IoC container. Rather than creating and managing dependencies within the object, dependencies are provided or "injected" into it, promoting loose coupling and modular design. DI reduces the coupling between components, making them easier to test, maintain, and modify. It enables flexible configuration and promotes reusability by allowing different implementations of dependencies to be easily swapped without modifying the dependent objects.

Q6.Explain the difference between constructor and setter injection?

Answer: Constructor injection and setter injection are two approaches for implementing dependency injection:

Constructor injection involves providing dependencies through the object's constructor. Dependencies are declared as constructor parameters, and the IoC container resolves and provides the dependencies when creating the object. This approach enforces the immutability of dependencies and ensures that the object is fully initialized upon creation.

Setter injection, on the other hand, involves providing dependencies through setter methods. Dependencies are declared as private fields with corresponding setter methods. The IoC container sets the dependencies by invoking the appropriate setter methods after object creation. Setter injection allows for flexibility in changing dependencies at runtime.

Q7.What are Spring Beans?

Answer: In the Spring Framework, beans are the objects managed by the IoC container. A bean is an instance of a class that is created, configured, and managed by the Spring container. Beans are defined in the Spring configuration file or through annotations. They represent the components of an application and can be wired together through dependency injection. Beans can have their lifecycle managed by the container, allowing for initialization, destruction, and various configuration options. Beans are the building blocks of a Spring application, providing modularity and flexibility.

Q8.What are the bean scopes available in Spring?

Answer: In Spring, there are several bean scopes available to define the lifecycle and visibility of beans:

1. Singleton: Default scope, where only one instance of the bean is created per container.
2. Prototype: A new instance of the bean is created each time it is requested.
3. Request: A new instance of the bean is created for each HTTP request.
4. Session: A new instance of the bean is created for each HTTP session.
5. Global Session: Similar to session scope, but for global (portlet-based) sessions.
6. Custom: Developers can define their own custom bean scopes as per their specific requirements.

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

Answer: Auto-wiring in Spring is a feature that automatically resolves dependencies between beans. It eliminates the need for explicit configuration by letting the Spring container automatically wire the dependencies based on certain rules. The different modes of auto-wiring are:

1. No: Default mode, where auto-wiring is disabled.

2. ByName: Dependencies are resolved based on the bean's name matching the property name.
3. ByType: Dependencies are resolved based on the type of the property. If there are multiple beans of the same type, an exception is thrown.
4. Constructor: Dependencies are resolved by matching constructor arguments with the beans defined in the container.
5. Autodetect: A combination of ByName and ByType, where auto-wiring is attempted by name first and then by type if necessary.

Q10.Explain Bean life cycle in Spring Bean Factory Container.
Answer: In the Spring Bean Factory Container, the life cycle of a bean consists of several stages:

1. Instantiation: The container creates a new instance of the bean.
2. Dependency Injection: Dependencies are resolved and injected into the bean.
3. BeanPostProcessor: Special callbacks are invoked before and after bean initialization.
4. Initialization: Initialization methods are called, allowing the bean to perform custom initialization tasks.
5. Ready for Use: The bean is now ready to be used by other beans or components.
6. Destruction: When the container is shutting down, the destruction methods of beans are called to perform cleanup tasks.

The BeanFactory Container manages and controls these stages to create, configure, and destroy beans in the application.