Assignment 9

Q1.What is Spring Framework?

ans)Spring Framework is an open-source Java framework that provides comprehensive infrastructure support for developing Java applications. It is designed to simplify the development process by providing features such as dependency injection, aspect-oriented programming, transaction management, and more. Spring Framework promotes modular and loosely coupled application development, making it easier to develop and maintain robust, scalable, and testable applications. It also offers integration with various other frameworks and technologies, making it a popular choice for building enterprise-level applications.

Q2.What are the features of Spring Framework?

ans)some of the features of the Spring Framework:

1. Dependency injection: Dependency injection is a design pattern that allows objects to obtain the dependencies they need from other objects. This makes it easier to test and maintain applications, and it also makes it easier to change the dependencies of an object without having to change the object itself.
2. Inversion of control: Inversion of control (IoC) is a design pattern that decouples the dependencies of an object from the object itself. This makes it easier to change the dependencies of an object without having to change the object itself.

3. Aspect-oriented programming (AOP): AOP is a programming technique that allows you to modularize cross-cutting concerns in your application. Cross-cutting concerns are things like logging, transactions, and security. AOP allows you to separate these concerns from the business logic of your application, making it easier to maintain and test your application.
4. Data access: The Spring Framework provides a number of features for data access, including JDBC, ORM, and JMS. These features make it easier to access data from a database or a messaging system.
5. Web development: The Spring Framework provides a number of features for web development, including MVC, WebSockets, and REST. These features make it easier to develop web applications that are scalable and easy to maintain.

Q3.What is a Spring configuration file?

ans)A Spring configuration file is a file that contains the configuration information for a Spring application. It is used to define the beans, their dependencies, and the way they are wired together.

Spring configuration files can be written in XML or Java. XML configuration files are the most common type of Spring configuration file. They are easy to read and write, and they are supported by most IDEs. Java configuration files are less common, but they offer more flexibility and control.

Q4.What do you mean by IoC Container?

ans)IoC (Inversion of Control) container, in the context of the Spring Framework, refers to a core feature that manages the creation, configuration, and lifecycle of application objects. It is responsible for applying the principle of inversion of control by controlling the flow of object creation and managing their dependencies. The IoC container eliminates the need for manual object instantiation and wiring of dependencies, allowing developers to focus on writing business logic. It achieves this by relying on configuration metadata, such as XML or annotations, which specify how objects should be created and how they should interact with each other. The IoC container is a key component in achieving loose coupling and enabling dependency injection in Spring applications.

Q5.What do you understand by Dependency Injection?

ans)Dependency Injection (DI) is a design pattern and a core principle in software development that promotes loose coupling and enhances modularity. It is a technique where the dependencies of a class or component are provided externally, rather than being created or managed within the class itself.

In Dependency Injection, the dependencies are "injected" into a class from the outside, typically through constructor parameters, method parameters, or setters. This allows for easier testing, reusability, and flexibility, as the class doesn't need to be aware of how its dependencies are created or managed.

By decoupling the class from its dependencies, Dependency Injection promotes better code organisation, modular design, and improves the overall maintainability and extensibility of an application. It also enables the use of different implementations or configurations of dependencies without modifying the class that depends on them.

Q6.Explain the difference between constructor and setter injection?

ans)

Constructor Injection	Setter Injection
Dependencies are provided through the class constructor.	Dependencies are provided through setter methods.
Dependencies are set at the time of object creation.	Dependencies can be set at any time after object creation.
Requires all mandatory dependencies to be passed during object creation.	Allows optional dependencies to be set separately.
Creates an immutable object with all dependencies set.	Allows modifying dependencies after object creation.
Promotes better encapsulation and ensures that all dependencies are available.	May result in a partially initialised object if not all dependencies are set.
Provides better control over object creation and	Provides flexibility in managing dependencies at

enforces dependency requirements.	runtime.

Q7.What are Spring Beans?

ans)Spring Beans are objects managed by the Spring Framework's IoC container. They are the fundamental building blocks of a Spring application and are instantiated, assembled, and managed by the container. Spring Beans are typically defined in the configuration files using XML or annotations.

Spring Beans represent various components of an application, such as services, data access objects, controllers, and more. They are responsible for providing specific functionalities and can be configured with various properties, dependencies, and behaviours. The IoC container handles the lifecycle of Spring Beans, including their creation, dependency injection, and destruction. Beans can be accessed and utilised by other components within the application.

Q8.What are the bean scopes available in Spring?

ans)In Spring Framework, the following bean scopes are available:

1. Singleton: This is the default scope. The container creates a single instance of the bean and reuses it whenever requested.

2. Prototype: A new instance of the bean is created every time it is requested from the container.

3. Request: A new instance of the bean is created for each HTTP request. This scope is applicable only in a web application.

4. Session: A new instance of the bean is created for each user session. This scope is applicable only in a web application.

5. Global Session: Similar to the session scope, but the bean is scoped to the global session in a portlet-based web application.

6. Custom Scopes: Spring also allows defining custom bean scopes as per the application's requirements.

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

ans)Autowiring in Spring is a feature that allows automatic dependency injection of beans into other beans without explicitly wiring them in the XML configuration or using Java annotations. It eliminates the need for manual wiring of dependencies, making the code more concise and less error-prone.

The different modes of autowiring in Spring are:

1. No autowiring (default): Dependencies are not automatically injected. Beans need to be wired explicitly using XML configuration or annotations.

2. ByName: Dependencies are autowired by matching the bean name with the property or constructor parameter name.

3. ByType: Dependencies are autowired by matching the bean type with the property or constructor parameter type. If multiple beans of the same type exist, an exception is thrown unless the **@Qualifier** annotation is used.

4. Constructor: Dependencies are autowired by matching constructor arguments with the beans in the container. It is similar to **ByType**, but applied to constructor arguments.

5. Autodetect: This mode is a combination of **ByName** and **ByType**. It first tries **ByName**, and if it fails, it falls back to **ByType**.

6. Annotation: Dependencies are autowired based on annotations like **@Autowired**, **@Inject**, or **@Resource**.

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

ans)In the Spring framework, the lifecycle of a bean managed by the Bean Factory container consists of the following steps:

1. Instantiation: The container creates an instance of the bean using the bean's constructor or a factory method.

2. Dependency Injection: The container injects the dependencies of the bean, either through constructor injection, setter injection, or field injection.

3. Initialization: If the bean implements the InitializingBean interface or defines an init-method, the container invokes the corresponding initialization callback method after the dependencies are injected.

4. Ready for Use: The bean is now fully configured and ready to be used by other beans or components.

5. Usage: The bean can be used by other beans or components to perform its designated tasks.

6. Destruction: If the bean implements the DisposableBean interface or defines a destroy-method, the container invokes the corresponding destruction callback method before the bean is removed from the container.