Q1. What is the Spring Framework?
The Spring Framework is an open-source application framework for Java that provides comprehensive infrastructure support for developing enterprise-level applications. It follows the principles of Inversion of Control (IoC) and Dependency Injection (DI) to promote loose coupling and modular development. Spring offers a wide range of features and modules that simplify the development of robust, scalable, and maintainable applications.

Q2. What are the features of the Spring Framework?
Some key features of the Spring Framework include:
- Inversion of Control (IoC) container for managing object dependencies.
- Dependency Injection (DI) for decoupling components and facilitating testing and reusability.
- Aspect-Oriented Programming (AOP) for modularizing cross-cutting concerns.
- Spring MVC for building web applications.
- Integration with various frameworks and technologies like Hibernate, JPA, JDBC, RESTful web services, etc.

Q3. What is a Spring configuration file?
A Spring configuration file is an XML or Java-based file that contains the configuration metadata for a Spring application. It defines how the Spring IoC container should create, configure, and wire the application's beans. The configuration file specifies the beans, their dependencies, properties, and other configurations required by the application.

Q4. What do you mean by IoC Container?
The IoC (Inversion of Control) container is a core component of the Spring Framework. It manages the creation, configuration, and lifecycle of objects (beans) in an application. The container, also known as the Spring container or Application Context, takes responsibility for instantiating and wiring the beans based on the configuration provided. It promotes loose coupling by allowing dependencies to be resolved and injected into the beans.

Q5. What do you understand by Dependency Injection?
Dependency Injection (DI) is a design pattern and a fundamental concept in the Spring Framework. It refers to the process of injecting the dependencies (i.e., collaborating objects or dependencies of a class) into a class from an external source rather than letting the class create the dependencies itself. DI helps in achieving loose coupling between classes and promotes easier testing, reusability, and maintainability.

Q6. Explain the difference between constructor and setter injection?
Constructor Injection and Setter Injection are two approaches for implementing Dependency Injection in Spring:

- Constructor Injection: In this approach, dependencies are injected through a constructor. The dependencies are declared as parameters in the constructor, and the Spring container resolves and provides the dependencies when creating the object. Constructor Injection promotes immutability and ensures that all required dependencies are satisfied before the object is created.

- Setter Injection: In this approach, dependencies are injected using setter methods. The dependencies are declared as private member variables in the class, and corresponding setter methods are provided. The Spring container calls the setter methods to inject the dependencies after creating the object. Setter Injection allows flexibility in terms of optional dependencies and supports bean reconfiguration.

Q7. What are Spring Beans?
In the Spring Framework, a bean is an object that is managed by the Spring IoC container. It is an instance of a class that is instantiated, configured, and assembled by the Spring container based on the bean's definition in the configuration file. Beans represent the various components and services of an application, and they are the building blocks that form the backbone of the Spring application.

Q8. What are the bean scopes available in Spring?
Spring provides several bean scopes that define the lifecycle and visibility of a bean. Some common bean scopes are:
- Singleton: The default scope, where a single instance of the bean is created and shared across the entire application context.
- Prototype: A new instance of the bean is created each time it is requested.
- Request: A new instance of the bean is created for each HTTP request in a web application.
- Session: A new instance of the bean is created for each HTTP session in a web application.
- Global Session: Similar to the session scope but used in a portlet context.

Q9. What is Autowiring, and name the different modes of it?
Autowiring is a feature in Spring that allows automatic dependency injection without explicitly specifying the dependencies in the configuration file. Spring automatically detects the dependencies required by a bean and attempts to satisfy them using the appropriate wiring strategy. The different autowiring modes in Spring are:
- byName: Autowiring is done by matching the bean name with the property name in the class.
- byType: Autowiring is done by matching the type of the bean with the property type in the class.
.

Q10. Explain the Bean lifecycle in the Spring Bean Factory Container.
The Spring Bean Factory Container manages the lifecycle of beans. The typical lifecycle stages of a Spring bean are:
- Instantiation: The container creates an instance of the bean by invoking the bean's constructor.

- Populating properties: The container sets the bean's properties using dependency injection or other means.
- Initialization: The bean's `init` method or any methods annotated with `@PostConstruct` are called, allowing the bean to perform initialization tasks.
- Usage: The bean is ready for use and can be accessed by other objects.