1. **What is the ApplicationContext ?**

One of the main features of spring is IoC container which is responsible for managing objects of application. The ApplicationContext represents the Spring IoC container, it provides basic functionalities for managing beans.

1. **What are the tradeoffs of different approaches to injecting beans ?**

Constructor injection- Pros: Clear dependencies, Ensures immutability. Cons: Can get into infinite loop by two dependent constructors.

Setter Injection- Pros: Init optional attributes not done in constructor(break loops), re-inject dependencies. Cons: Allows partially initialized beans.

Field Injection- Pros: Clean syntax. Cons: Mutable, hard to test, hides dependencies.

1. **Why do we need to use @Qualifier when multiple of the same type are defined ?**

Spring will not know which one to use and it will cause errors.

1. **How to avoid loading of heavy beans (like caches or other beans with heavy init logic) on startup and decrease startup time?**

We can do it by using lazy initialization. A lazy-initialized bean tells the IoC container to create a bean instance when it is first requested, rather than at startup.

1. **What are Spring lifecycle stages and methods?**

* Container Started: The Spring IoC container is initialized.
* Bean Instantiated: The container creates an instance of the bean.
* Dependencies Injected: The container injects the dependencies into the bean.
* Custom init() method: If the bean implements InitializingBean or has a custom initialization method specified via @PostConstruct or init-method.
* Bean is Ready: The bean is now fully initialized and ready to be used.
* Custom utility method: This could be any custom method you have defined in your bean.
* Custom destroy() method: If the bean implements DisposableBean or has a custom destruction method specified via @PreDestroy or destroy-method, it is called when the container is shutting down.