*Main repository:*

<https://github.com/Shiv-Sunder-Dash/nisum_sprint2/tree/main/spring/assignment>

Xml:

<https://github.com/Shiv-Sunder-Dash/nisum_sprint2/blob/main/spring/assignment/pom.xml>

<https://github.com/Shiv-Sunder-Dash/nisum_sprint2/blob/main/spring/assignment/src/main/resources/beans.xml>  
java folders and files:

<https://github.com/Shiv-Sunder-Dash/nisum_sprint2/tree/main/spring/assignment/src/main/java/com/nisum/assignment>

<https://github.com/Shiv-Sunder-Dash/nisum_sprint2/blob/main/spring/assignment/src/test/java/com/nisum/assignment/AssignmentApplicationTests.java>

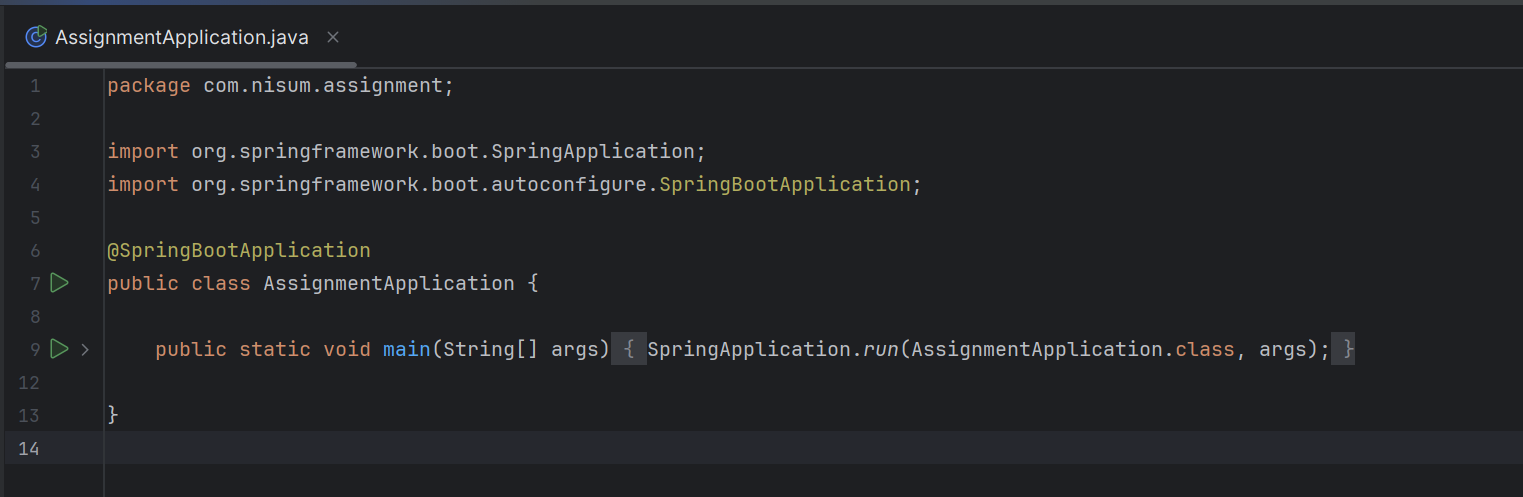
Application properties:

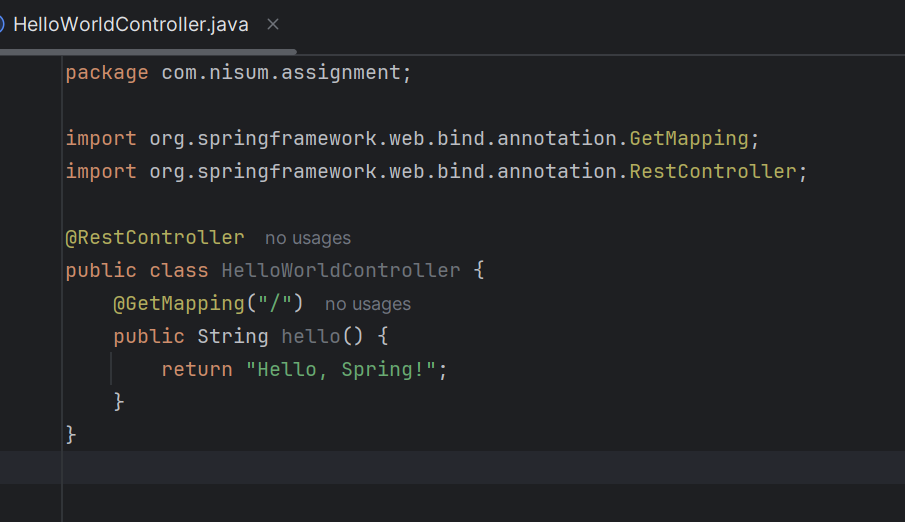
<https://github.com/Shiv-Sunder-Dash/nisum_sprint2/blob/main/spring/assignment/src/main/resources/application.properties>

*Introduction to Spring Framework : Assignment:*

1 . I have done the reading

2 .







3 a. The purpose of the Spring Framework is to simplify enterprise Java development by providing infrastructure support for building robust and maintainable applications. It offers features like dependency injection to reduce tight coupling between components, aspect-oriented programming to separate cross-cutting concerns such as logging and transactions, and provides integration with various technologies like JDBC, JPA, JMS, and more. It also supports building web applications through its MVC module and promotes modular development where only required components are used.

3 b. The main modules of the Spring Framework are:

Core Container (includes Core, Beans, Context, and Expression Language)

Spring AOP (Aspect-Oriented Programming)

Data Access/Integration (JDBC, ORM, JMS, Transactions)

Web (Web, Web MVC, Web WebSocket, Web Reactive)

Test (unit and integration testing support)

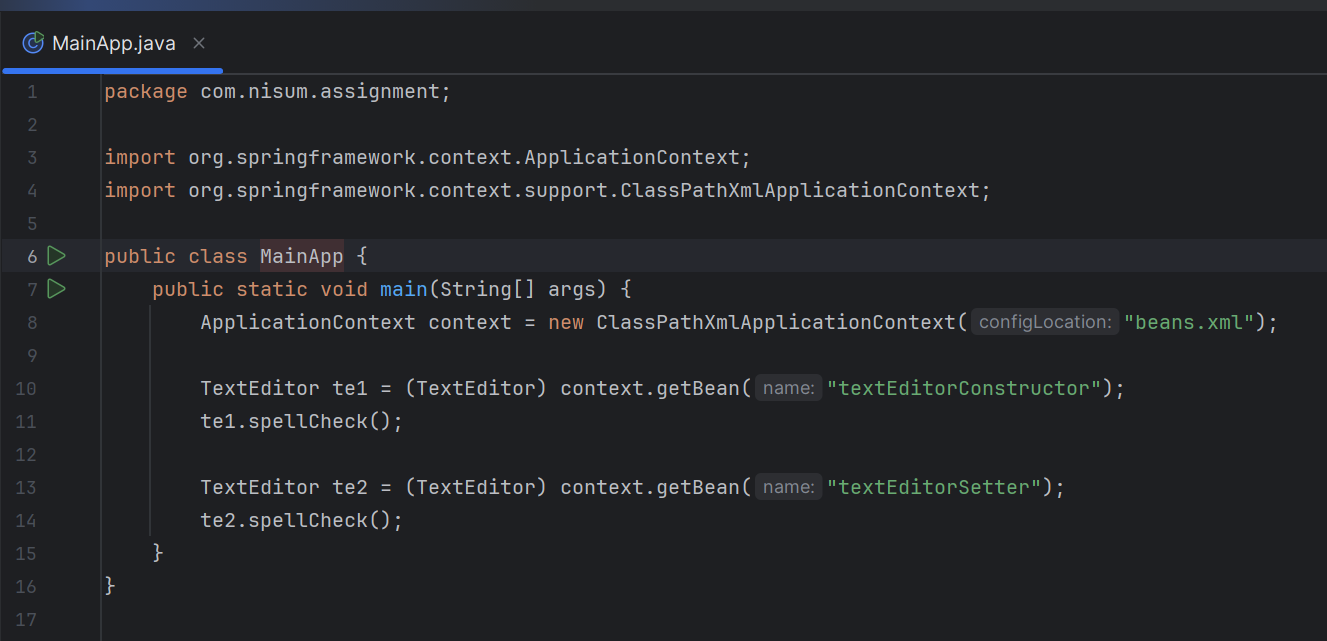
*Core Concepts : Assignment:*

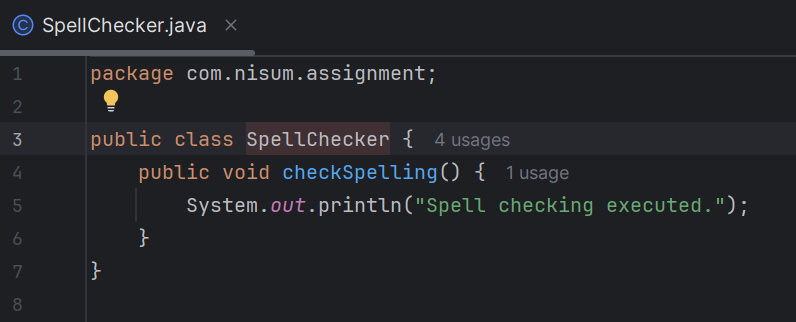
1. Reading:

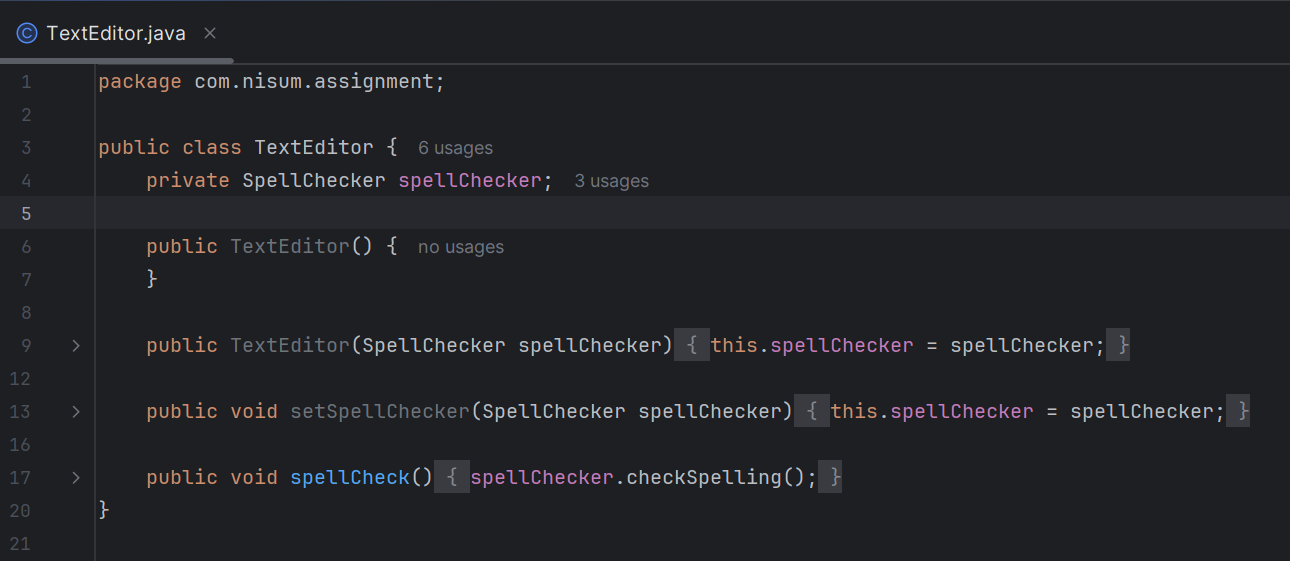
Dependency Injection (DI) is a design pattern used in Spring to achieve loose coupling between classes by injecting dependencies from the outside rather than creating them internally. Inversion of Control (IoC) refers to the principle where the control of object creation and lifecycle is transferred from the application code to the Spring container.

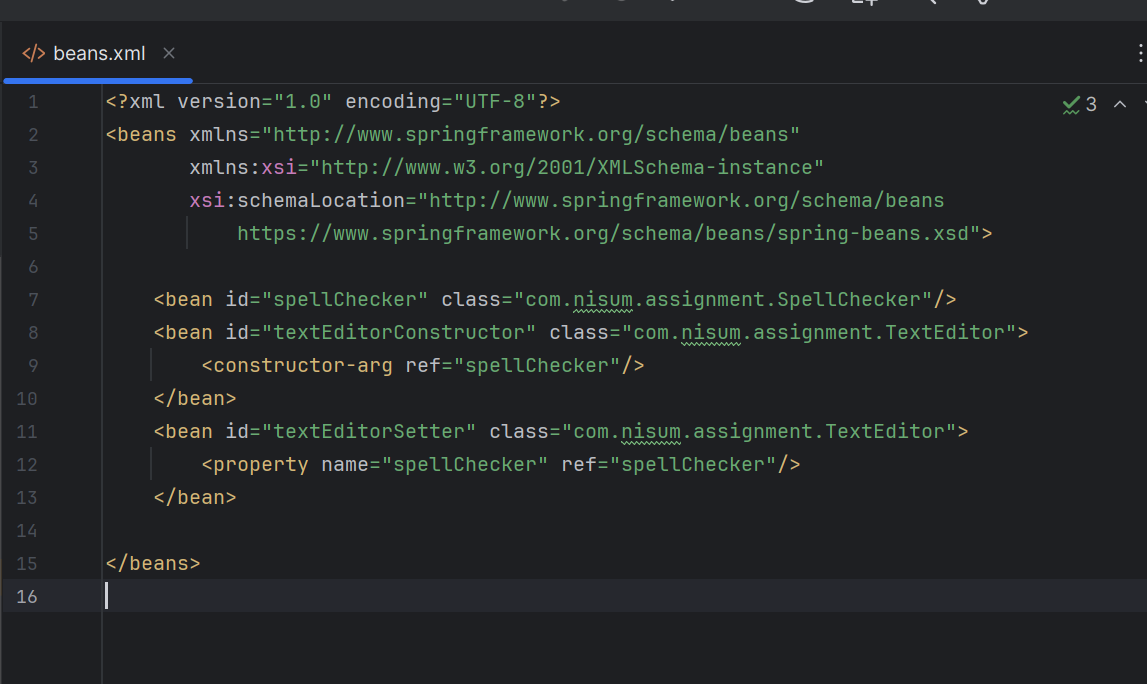
BeanFactory is the basic container in Spring that provides fundamental support for managing beans. It lazily loads beans and is suitable for simple applications. ApplicationContext is a more advanced container that eagerly loads beans and provides additional features like internationalization, event handling, and automatic bean registration through annotations.

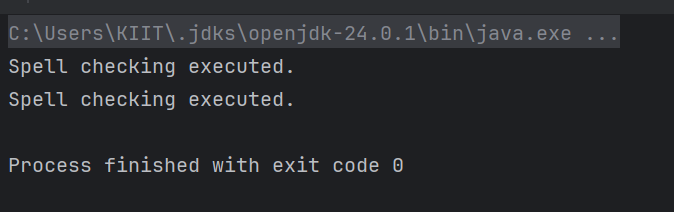
2.











3. What is Dependency Injection, and why is it important?

Dependency Injection (DI) is a design pattern where an object receives its dependencies from an external source rather than creating them itself. In Spring, the container injects dependencies into beans. This is important because it promotes loose coupling, better code maintainability, and easier testing by removing the need for manual object creation and wiring.

Explain the difference between Constructor Injection and Setter Injection.

Constructor Injection: Dependencies are provided through a class constructor. It ensures that the dependency is available at the time of object creation and is good for mandatory dependencies.

Setter Injection: Dependencies are provided using setter methods after object creation. It allows for optional dependencies and more flexibility but risks leaving objects partially initialized if not used carefully.

What is the difference between BeanFactory and ApplicationContext?

BeanFactory is the basic Spring container, offering lazy initialization and fundamental bean management.

ApplicationContext is a more advanced container that builds on BeanFactory. It supports features like internationalization, event propagation, AOP integration, and automatic bean registration using annotations.

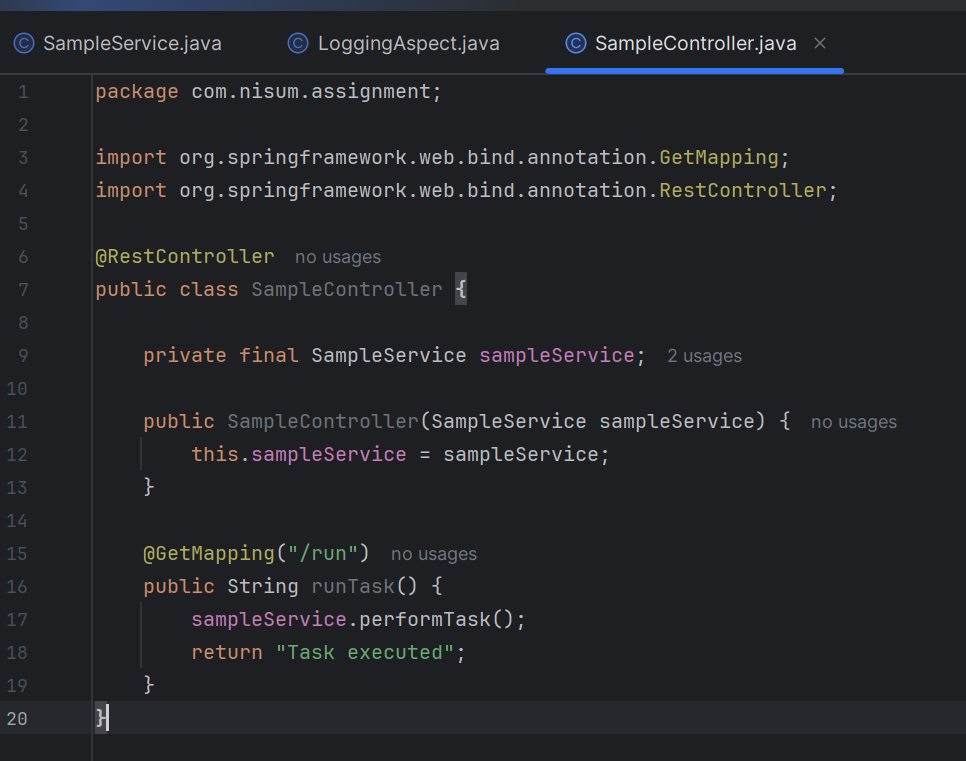
How does Spring manage the lifecycle of beans?

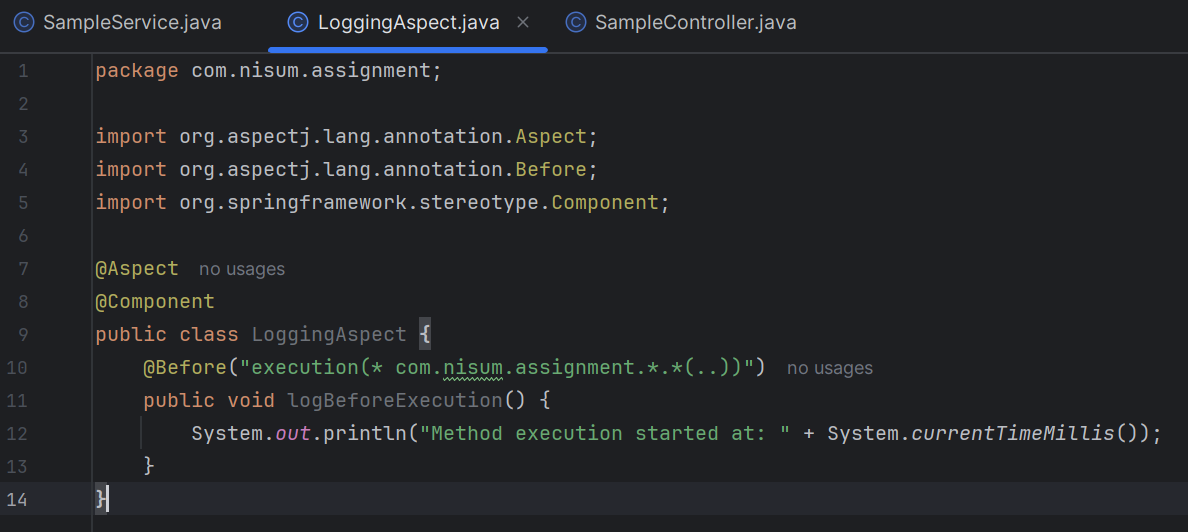
Spring manages the lifecycle of beans through the container. It creates beans, injects dependencies, calls initialization methods (@PostConstruct or init-method), keeps beans in memory as per their scope (singleton, prototype, etc.), and optionally calls destroy methods (@PreDestroy or destroy-method) when the application context is closed.

*Aspect-Oriented Programming (AOP) : Assignment*

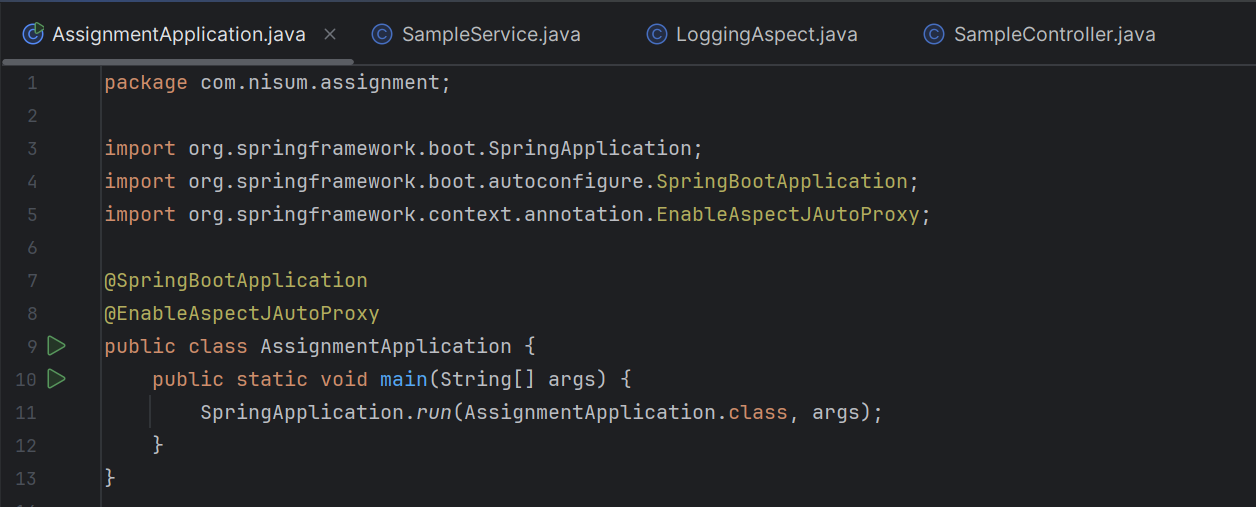
1. Reading:  
   Aspect-Oriented Programming (AOP) in Spring is a programming technique that allows the separation of cross-cutting concerns like logging, security, and transaction management from the main business logic. It helps to reduce code duplication and improve modularity by enabling the definition of aspects, which can be applied declaratively to various parts of the application using annotations or XML configuration

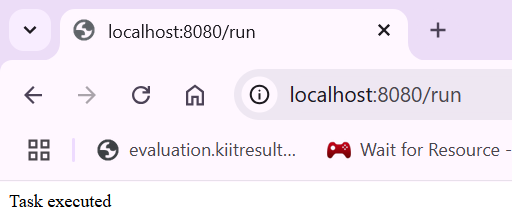
2.

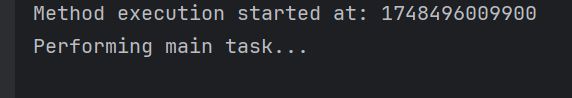












3. Questions:

Aspect-Oriented Programming (AOP) is a programming paradigm that allows separation of cross-cutting concerns such as logging, security, or transaction management from the main business logic. In Spring, AOP is used to define common behaviors (aspects) and apply them across multiple classes without modifying their source code. This is achieved using annotations like @Aspect, @Before, @After, and proxies managed by the Spring container.

The different types of advice in AOP are:

@Before – Runs before the target method execution

@After – Runs after the method completes (whether it succeeds or fails)

@AfterReturning – Runs only if the method completes successfully

@AfterThrowing - Runs only if the method throws an exception

@Around – Wraps the method execution; used to measure execution time or alter behavior before and after the method runs

Fill in the blanks and MCQ:

1. byName, byType, constructor, autodetect (deprecated), no (default).

2.It is an XML tag used to activate annotation-based dependency injection (like @Autowired, @PostConstruct, etc.) in Spring configuration files.

3.(d) @EnableAspectJAutoProxy

4.(c) aop:aspectj-autoproxy

5.(d) All of the above

6.(b) FALSE

7.inner beans

8.(c) ContextLoaderListener

9.(b)Transaction management

10.(b)ApplicationContext

11.(a)The setter method becomes optional

12.(d) Field Injection