Created by:-Lalit Ingale

Spring Core

1)What is Spring?

➔Spring is an open source java framework.

-Using spring we can develop Standalone&Enterprise application.

-Spring was initially release 2003.

-Spring was production release(1.0) in 2004.

-Spring was developed by “Rod Johnson”.

2)Which are the Advantages of Spring?

➔**i)Modular and Lightweight**(eaning you can use only what you need, and lightweight, meaning it adds minimal runtime overhead.)

ii)Flexible Configuration(Flexible configuration in Spring allows developers to choose between XML, annotations, or Java-based setup)

iii)Dependency Injection(Dependency Injection (DI) in Spring promotes loose coupling and easier testing by allowing objects to be injected rather than created manually within classes)

iv)Aspect-Oriented Programming(Allows separation of cross-cutting concerns like logging and security, leading to cleaner and more maintainable code)

v)Simplified Database Access(through JDBC and ORM support, reduces boilerplate code and streamlines data operations for faster and cleaner development)

vi)Testing Support(Spring architecture encourages write unit tests, and it provide support for integration testing)

vii)Security(Spring Security provides comprehensive authentication and authorization, helping protect applications with minimal custom code)

viii)Integration Capabilities(Spring provides a integration with various technologies and framework, such as angular,react,messaging system(JMS),web service(SOAP and REST),and other third party libraries and APIs)

ix)Scalability(Spring application can be designed for scalability and can easily integrate with cloud-native technologies and microservice architecture)

x)Open Source(Means it's free to use and can be customized to meet specific project requirement)

3)What POJO(Plain Old Java Object) Class?

➔A simple java class with fields and getters/setters, used for data representation without framework dependencies.

4)What is JavaBean Class?

➔A Serializable class with a no-arguments constructor, often used to encapsulate data and adhere to JavaBeans convention for easy integration with framework.

5)Which jar files are needed to spring configuration file xml schema?

➔i)Spring-**beans**-xxx.jar

ii)Spring-**core**-xxx.jar

iii)Spring-**context**-xxx.jar

iv)Common-**logging**-xxx.jar

v)Spring-**expression**-xxx.jar

6)Define @Component?

➔The "@Component" annotation in Spring is used to declare a class as a spring bean, which is a managed component in the spring application context.

-It helps spring automatically detect and manage these beans during application startup, making them available for dependency injection and other spring feature.

7)Define @Value?

➔The @Value annotation in spring is used to inject values in spring bean fields or methods.

-@Value is mostly used to inject values from external sources(e.g, properties files or environment variable)

8)Which annotation are based on "@Component" annotation?

➔**i)@Service:-**It handles the business logic like calculation and validation.

ii)@Repository:-Marks a class as DAO(Means talk to database)

iii)@Controller:-Marks a class as Controller, Which means it handle web request.

9)Define @ComponentScan?

➔@ComponentScan annotation is used to instruct the spring framework to perform component scanning and then register them as spring beans for further use.

10)What is JavaBean Object?

➔A "Java Bean Object" is an instance of JavaBean Class.

-It is reusable,encapsulated java component with properties, getter, and setters,designed for easy integration and manipulation.

11)Define Life Cycle of Bean-Object?

➔**i)Loading Bean Defination(**Spring loads bean definition from various sources, such as XML configuration files, Java based Configuration classes or Component Scanning)

ii)Bean Instantiation(After Bean definition is loaded, the Spring container creates instances of beans based on these definition.

-This involves invoking the bean class's constructor, either the default constructor or specified constructor, to create an actual instance of the bean.

-The newly created bean instance is now ready for configuration and initialization.

iii)Bean Initialization(Once the bean is instantiated, Spring proceeds to config and initialize it)

-Property values are set using Setters, constructor arguments or field injection, populating the bean's state.

12)What is BeanPostProcessor?

➔A “BeanPostProcessor” in Spring is an interface that enables custom logic to be executed before and after the initialization of spring beans.

13)Define ref attribute?

➔ref attribute is used to inject other one object value to another.

14)Define constructor-arg?

➔ In spring framework, constructor-arg is used in XML configuration to pass values to a constructor when spring creates an object(bean).

15)Define getBean()?

➔In Spring Framework, getBean() is a method used to get or create an object(bean) from the Spring Container.

16)What is Bean in Spring?

➔A bean in spring means, An object created and managed by the framework.

17)Define @Configuration?

➔@Configuration in Spring marks a class as providing bean definition for the application context, enabling java based configuration.

18)Define @Bean?

➔@Bean is Spring marks a method to produce a bean object which is managed by the spring container.

19)Define AutoWiring?

➔It is the feature of Spring framework by which we can achieve "DI automatically".

- **Auto** ↓ "Automatically" manages The connection between Objects

Wiring ↓ Linking those objects together to fulfill dependencies.

20)Define @Autowired?

➔-The @Autowired annotation in spring is used for automatic dependency injection.
-It tells the Spring: "Hey, just grab the right piece(bean) and plug it in here to make this work.
-In short, it simplifies your code by letting spring handle object connection for you.

21)Define @Qualifier?

➔The @Qualifier annotation in Spring helps pick the right bean among multiple beans of the same type, It helps spring to know which bean you want injected, resolving ambiguity.

22)What is the difference between @Autowired and @Qualifier annotation?

➔ @Autowired is used to automatically inject a dependency by type.
If there are multiple beans of the same type, @Qualifier is used along with @Autowired to specify which exact bean to inject by name.

23)What is Autowired Modes?

➔-These modes define how strictly Spring should try to inject the object.

-There are 3 main modes.
i)ByName:-If your property is named is acr, Spring looks for a bean named car.
ii)ByType:-If the property is of type car, Spring looks for a car type bean.
iii)Constructor:-Spring looks at constructor parameter and trying to inject matching beans.

24)Explain autowire-candidate?
➔In Spring, autowire-candidate means:"Can this bean be chosen for autowiring or not?"
-Think of it like a yes/no flag you set on.
-"autowire-candidate=false"means:
Don't use this bean when spring is trying to autowire dependencies.
-"autowire-candidate=true"(default value) means:

25)What is MAVEN?
➔-It is a build tool which automates everything related to the building of project.
-Maven was developed by JASON VAN ZYL in 2004(Apache software foundation)

26)Define the Responsibilities of Maven?
➔i)Creates the Project Structure
ii)Download the required dependencies(jar files)
iii)Prepares the documentation
iv)Compile Source Code
v)Starts or Stop the Server
vi)Packaging the project in JAR or WAR or EAR file.

27)What is SPRING JDBC MODULE?
➔We can connect spring application with database using JDBC and for

this spring framework provides one module i.e.”Spring JDBC Module”

28)Which classes provide Spring JDBC Module?

➔**i)DriverManagerDataSource**----Provide Configuration
ii)JDBC Template-----Provide CRUD Operation
iii)NamedParameterJdbcTemplate

29)What is “NamedParameterJdbcTemplate” class?

➔-NamedParameterJdbcTemplate is a Spring class for executing SQL queries with named parameter instead of “?” placeholders.
-It improves readability by replacing positional placeholders with named parameter, enhancing code maintainability and reducing errors in SQL queries.

30)What is Design Pattern?

➔-Design patterns are ‘Reusable Solution to common design problems’…
-enhancing code structure, scalability, and maintainability in web development.
-Examples are:
i)Singleton Design Pattern
ii)Factory Method Design Pattern
iii)DAO Design Pattern
iv)MVC Design Pattern

31)Define MVC Design Pattern?

➔M---**Model**(Used to store ‘Data’)
V----**View**(Used for ‘Frontend’ or ‘Presentation’)
C----**Controller**(Used for ‘Backend’)
➔MVC design pattern separates the different concerns i.e.’Data Layer’, ‘Presentation Layer’ and ‘Buisness Layer’ in order to develop organized and modular java application.

32)What is Spring WEB Module?

➔Spring Web Module is the part of spring framework that provides the foundation for creating web application.
-It offers features like handling http requests and responses, integrating with various web technology managing session etc…

33)What is Spring WEB-MVC Module?

➔-Spring WEB-MVC is a specific module within spring web that implements the Model-View-Controller(MVC) Design Pattern.
-It simplifies building web applications by separating the Data Layer(Model), Presentation Layer(View) and Buisness Logic Layer(Controller)

34)Define @RequestParam?

➔ @RequestParam is used in Spring MVC to get values from the URL query string or form data and pass them into your controller method.

35)Define Model?

➔ Model is like a container that holds your data so that it can be shown in the View (like an HTML page).

36)Define @ModelAttribute?

➔This annotation are used to take the data for form, and store in object(model) Automatically.

37)Which are the modules used in Spring Core?

➔The modules names are core,web mvc,jee,aop,security,data jpa,data mongodb,cloud…..

38)Which are the IOC container/Spring containers are used in all modules?

➔**i)BeanFactory(Basic)**

- BeanFactory is the most basic IoC container in Spring.
It is responsible for creating and managing beans (objects), but only when they are needed (lazy loading).

ii)ApplicationContext(Advanced)

- ApplicationContext is the advanced IoC container in Spring that not only creates and manages beans but also provides extra features like:

Automatic bean creation at startup,,, Internationalization (i18n) support,,,

Event handling,,, Bean lifecycle management.

➔**IOC container:-** The IoC Container in Spring is a framework component that automatically creates, configures, and manages the lifecycle and dependencies of objects (beans) in an application.

39)What is Spring Bean Life Cycle?

➔Spring Bean Life Cycle management means that IOC container takes care of spring bean life cycle end to end(loading class, creating object, managing beans, calling life cycle methods and destroying the object)

40)What is Spring Bean Dependency Management?

➔Spring Bean Dependency management means that the IOC container arranges one Spring bean class obj(Dependent obj) to another spring bean class object(target object)

41)What is Spring Bean?

➔The java class whose objects is created and managed by IOC container is called Spring Bean.

42)What is Configuration?

➔The process of giving inputs and instruction to IOC container is called configuration.

43)Which are the 3 approaches are used to give inputs and instruction to the spring programming?

➔**a)using xml driven cfgs(**all inputs and instruction will be given to IOC container using xml file)

b)using xml+annotation driven cfgs(first the inputs and instruction will be given using annotation,if not sufficient then we go for xml file)

c)using 100% java code configuration(All inputs and instruction will be given using both annotation and java statement i.e xml file will be eliminated or minimized).

44)What is spring bean cfg file?

➔The xml file using which we provide inputs and instruction to IOC container is called spring bean cfg file.--------applicationContext.xml

45)What is Configuration class?

➔The java class using which we provide inputs and instruction to IOC container is called Configuration file.---------AppConfig.java

46)What are the Advantages of annotation?

- These are the java statements,so can be kept in java class itself.
- The compiler can recognize the annotation directly.
- The JRE/JVM,containers/servers/frameworks can process the annotations directly to provide respective functionality.
- Improves the readability of the code becoz they are placed directly in java code.
- Suitable for the java code configuration and industry standard.

47)Can'we use @Component Annotation to cfg pre-defined class as spring bean?

➔No,not possible becoz @Component is class level annotation and we cannot place that annotation on the top of pre-defined classes as we can not open the source code of pre-defined classes.

➔to configure and pre-defined class as the spring bean we need to use @Bean method support in @Configuration class.

48)Can use @Bean method support to configure user-defined java class as spring bean?

➔Yes, but not recommended becoz we need to create and return obj for user-defined class explicitly in process.. So prefer using @Component annotation to make the user-defined java class as the spring bean.

49)How to creating BeanFactory IOC container?

➔To create this container, take a java class that implements the spring api supplied <u>BeanFactory(I)</u> directly or indirectly.

➔Through there are multiple classes implementing BeanFactory(I), the mostly used class to create BeanFactory IOC container is "XmlBeanFactory".

50)What are limitation of BeanFactory IOC container?

- Offering less features then ApplicationContext IOC container.
- Does not support annotation,java code driven configuration.
- We need to use separate resource obj to locate and hold spring bean cfg file.
- Lots of mutual configuration are required.
- Not a industry standard IOC container.

51)Which are the most important classes are used to create ApplicationContext container?

➔a)FileSystemXmlApplicationContext

b)ClassPathXmlApplicationContext / **Supports only xml driven cfgs**

c)XmlWebApplicationContext

d)AnnotationConfigApplicationContext \ **supports annotation**

e)AnnotationConfigWebApplicationContext

52)Using FileSystemXmlApplicationContext.

➔Creates the ApplicationContext IOC container by locating the given spring bean cfg file from the specified file system.

➔FileSystemXmlApplicationContext ctx = new FileSystemXmlApplicationContext("name and location of spring bean cfg file");

53)Using ClassPathXmlApplicationContext.

➔Creates the ApplicationContext IOC container by locating the given spring bean cfg file from the directories and jar files added to the CLASSPATH.

➔ClassPathXmlApplicationContext ctx = new ClassPathXmlApplicationContext("name of the spring bean cfg file that will be located from the directories and jar files added to the classpath")

54)Using XmlWebApplicationContext.

➔Useful to create IOC container in web application by taking <servlet logical name>servlet.xml file as the spring bean cfg file from the WEB-INF folder of the web application.

➔XmlApplicationContext ctx = new Xml<u>ApplicationContext</u>();

(if the servlet comp logical name is "controller"

then it takes WEB-INF/controller-servlet.xml file

as the spring bean cfg file)

55)Using AnnotationConfigApplicationContext.class.

➔Useful to create ApplicationContext IOC container in standalone apps by taking given java class as the configuration class as alternate to xml file.

➔AnnotationConfigApplicationContext ctx = new annotationConfigApplicationContext(AppConfig.class);

@Configuration Class

to give input and instruction

to the IOC container.

56)Using AnnotationConfigWebApplicationContext.

➔This is very useful to create IOC container in web application setup by taking given java class as the @Configuration class.

➔AnnotationConfigWebApplicationContext ctx = new AnnotationConfigWebApplicationContext(AppConfig.class);

57What is Dependency Management?

➔It is the process of arranging dependent spring bean for target spring bean so the target spring bean can use the services of dependent spring bean.

➔Target/main spring bean___dependent/helper spring bean

Flipcart	DTDC(Flipcart uses DTDC courier services for delivering products)

58)In Spring/SpringBoot the dependency management can be implemented in two ways.

➔**i)Dependency Lookup:-**If the target spring bean class is searching and getting dependent spring bean from various other resources of the app/project then it is called dependency lookup.

-The way java app/java class(Target) is getting required obj ref(dependent)

by searching in jndi registry is called Dependency Lookup.

JNDI:: Java Naming And Directory Interface

-If want to provide global visibility and accessibility to any object then place that object ref in the Jndi registry s/w.----RMI registry,,COS registry.

59)What is Dependency Injection?

➔If the underlying server/container/framework/JVM/…dynamically assign dependent spring bean class obj to target spring bean class obj then is called dependency Injection.

➔The way ServletConfig objects is injected to our servlet class obj by ServletContainer.

$$\begin{tikzcd} \text{ServletConfig} \arrow[d] & \text{servlet} \arrow[d] \\ \text{(dependent)} & \text{(Target)} \end{tikzcd}$$

60)What are the advantages of Dependency Injection?

➔Here the underlying server/container/framework/JVM dynamically assign Dependents to target class obj_No burden to the programmer.

➔Programmer need not to write logics in target class to get dependents.

➔In 90% cases we prefer working with Dependency Injection, in 10% cases we prefer to working with Dependency Lookup.

61)Dependency Management in Spring.

➔ Dependency Management in Spring

Dependency Lookup(DL)

==================

i)For this we need to call getBean(-) on IOC container obj passing the bean id

Dependency Injection(DI)

=====================

This can be implemented in multiple ways by using xml driven cfgs or annotation driven cfgs.

➔As setter injection

➔As constructor injection

➔As arbitrary method injection

➔As filed injection

Basic Injection

➔Interface Injection

➔Lookup Method Injection

➔Method Injection

Advanced Injection

62)What is Setter Injection?

➔If the IOC container using setter method of target spring bean class to assign the dependent spring bean class obj then it is called setter injection.

63)What is Constructor Injection?

➔If the IOC container is using parameterized constructor of target spring bean class to create the target spring bean class and also to assign dependent spring bean class obj then it is called constructor injection.

64)What is arbitrary method Injection?

➔If the IOC container is using arbitrary method(random method with any name) of target spring bean class to assign dependent spring bean class obj to target spring class obj is called arbitrary method injection.

65)What is Field Injection?

➔If the IOC container is using Field/HAS-A property of target spring bean class to assign dependent spring bean class object to target spring bean class object then it is called Field Injection.

66)Which tags are used to inject that property in xml driven cfgs?

➔In xml driven cfgs we use:

a)**<property>** tags under <bean> tag for setter injection cfgs.

b)<**contructor-arg>** tags under <bean> tag for constrctor injection cfgs.

67) Which tags are used to inject that property in Annotation or java code driven cfgs?

➔In Annotation or java code driven cfgs,we use @Autowired annotation for all the 4 injection.

➔wiring=injection,So @Autowired gives to instruction to IOC container to perform automatic injection to target spring bean class.

68)What is XML?

➔XML(Extensible Markup Language) is a flexible, structured format used to store and transport data, It is both human-readable and machine-readable.

69)If we configure both setter injection and constructor injection on the same spring bean property with two different values then we which injection values will be taken as the final values?

➔Since the setter method executes after constructor, so we can say the value injected by the constructor injection will be overridden with the value given by the setter injection.

70)What is Cyclic Injection?

➔If two spring beans are dependent to each other then we can say they are in Cyclic Dependency Injection.

71)What is the difference between setter injection and constructor injection?

➔

setter injection	**constructor injection**
i)IOC container uses setter method to assigning Dependent spring bean class obj to target spring bean class object.	i)IOC constructor uses param-constructor to create target spring bean class obj and to ass-gn Dependent spring bean class obj to target

-first create TSBCO and then DSBCLO.	spring bean class obj.
ii)For this, use <property> tag in xml driven cfgs.	ii)For this use <constructor-arg> tag in xml driven cfgs.
iii)For this, use @Autowired on the top of the setter method in annotation driven,java code driven cfgs.	iii)For this, use @Autowired on the top of param-constructor in the annotation driven cfgs.
iv)Supports Cyclic or Circular DI.	iv)The IOC creates dependent spring bean
v)For this injection IOC container uses 0-param-constructor to create target spring bean class object.	class obj first then it creates target spring target spring bean class obj having dependent spring bean class obj as the constructor
vi)Bit slower injection compare to constructor injection.	arg value. v)Does not support Cyclic or Circulare DI.
vii)Good to use if the property of spring bean are optional to configure for injection.	constructor to create target spring bean class obj.
viii)To perform setter injection cfgs on our choice no, of spring bean properties in all permutation and combination we need to take max of n setter method for n properties. (eg-4 properties we need 4 setter methods)	vii)Faster injection then setter injection. viii)Good to use if the properties of spring ix)To perform constructor injection, (eg 4 properties we need(4*3*2*1) no of con.

72)If we configure multiple dependency injection on the same spring bean property having different values then which injection value will be taken as the final value?

➔If arbitrary method is placed after setter method then arbitrary method injection value will be taken as the final value. If the setter method is placed after arbitrary method then the setter injection value will be taken final value.

73)Define @Primary annotation?

➔In Spring Framework, @Primary is an annotation used to tell Spring which bean to use by default when there are multiple beans of the same type.

74)If we apply all the four injection(field injection,setter injection,constructor injection and arbitrary method injection) on single Spring bean property of target spring bean class having four different spring bean objs of same type,,then which injection will be taken as the final injection?

➔If setter method of setter injection is placed after arbitrary method of arbitrary method injection then setter injection value/object will be taken as the final/value object.

➔If arbitrary method of arbitrary injection is placed after setter method of setter injection then arbitrary method injection value/object will be taken as the final value/objects.

75)Why @Qualifier is best solution to solve the ambiguity problem?

➔The reason are:

a)The bean id required in @Qualifier(-) can be gathered from properties file or xml file to make the code as loosly coupled code.

b)If we apply multiple solutions(@Primary,@Qualifier(-),name matching) on the single HAS -a property of a target spring bean to solve the ambiguity problem…the @Qualifier(-) solution will be taken as the final solution.

76)What is the difference between FileSystemXmlApplicationContext and ClassPathXmlApplicationContext towards creating IOC container in xml driven cfgs based Spring Application Development?

➔FileSystemXmlApplicationContext:-

-This class creates the IOC container of type ApplicationContext by locating the given spring bean file from the specified path of file system, So we can pass either relative path(respect to project location) or absoulate path(right from root folder) of spring bean cfg file.

➔ClassPathXmlApplicationContext:-

-This class creates ApplicationContext IOC container by locating the given spring bean cfg file from the folders and jar files added to the CLASSPATH/BUILDPATH of the project.

-By default "src" folder of the Eclipse project is placed in the CLASSPATH or BUILDPATH of project automatically.

77)What is Design Patterns?

➔-Design patterns are bunch of rules that acts as best solutions for reoccurring problem of Application Development.

-Design patterns are best practice to use programming languages, technologies and frameworks effctively in application development.

a)GOF/GO4 Design Patterns

b)JEE Patterns

78)What is Layered application development?

➔Keeping different category logics in different classes or files and making them interacting with each other is called Layered application development.

79)What is Factory Method?

➔Any method that creates and returns the object ic called Factory Method.

80)What is the difference between Architecure and Design Pattern?

➔-Architecture refers to the overall structure of Spring—like its layers (Core, AOP, Data Access, MVC, etc.) and how components like beans, context, and dependency injection work together.

-Design patterns are the reusable coding techniques Spring uses internally—like Singleton, Factory, Proxy, and Template Method—to solve common software problems.

81)What is the difference between factory method and factory patterns?

➔-Factory method creates and returns any class obj(current class or related class or unrelated class)

-Factory patterns internally uses factory method but returns one of the several related classes obj(based on the data that is passed)

82)Which are the Design Patterns are used in java domain?

➔**i)GOf/GO4 Patterns**(e.g-singleton class, factory, abstract factory and etc)

ii)JEE Patterns(e.g-DAO, FrontController, composite view and etc..)

iii)MicroServices Patterns(e.g-API Gateway, Circuit Breaker, SAGA and etc)

83)Define @Lazy?

➔@Lazy means "Don't create this object now — wait until it's really needed."

84)Define @Scope?

➔-@Scope tells Spring how long an object (bean) should live and how it should be shared.

-It decides whether to create one object for the whole app, or a new object every time someone asks for it.

85)What is singleton java class?

➔The java class that allows us to create only one object in any situation is called Singleton java class.

86)What is reflection API?

➔A part of Java that lets your program look into and interact with classes, methods, and variables while the program is running — even if their details were not known at compile time.

87)Can we place only private constructor(s) in spring bean class?

➔yes, we can place becoz the IOC container uses reflection API internally to get access to private constructor and to create the spring bean class object.

88)Can we take spring bean class as the private class?

➔Java does not allow to take outer classes as private classes, It allows to take only inner classes as private classes. Since we do not take inner classes as spring beans we can say overall the spring class can not be taken as the private class.

89)What are two special properties that the singleton scope spring bean is having compare to other scope spring beans?

➔ A singleton scope Spring bean has only one instance for the entire application and is created at startup, unlike other scopes where a new instance is created when needed.

90)How can we get singleton class effect on spring bean without making the spring bean class as singleton class?

➔ @Scope("singleton") - Spring ensures only one bean instance is created and shared, even if the class allows multiple objects.

91) Define @Scope(Prototype)?

➔-By default, Spring creates only one object (called a bean) of a class and reuses it every time (this is called singleton scope).

-But when you use @Scope("prototype"), Spring will create a new object every time you ask for the bean.

92)What happen if we configure the real singleton java class as the prototype scope spring bean?

➔If IOC container is creating singleton java class obj by accessing private constructor then the prototype scope will be continued i.e the IOC container creates the multiple objs for multiple ctx.getBean() method class(Singleton java class will be broken)

93)If we do not provide bean id to the spring bean class then what happens?

➔The IOC container generates default bean ids to the spring bean class based on the approach that we have used to configure the spring bean class.

94)What is the difference between IOC and dependency Injection?

➔IOC is the software specification having set of rules and guidelines to manage the dependency among the comps(like target and dependent spring beans)..”Dependency Lookup”, “Dependency Injection” are the two implementation method that are given based on IOC rules and guidelines.

95)When should i take the spring bean scope as the prototype scope?

➔If the data of the spring bean is changing time to time (like storing from data to the spring bean class obj) then we need to take that spring bean as prototype scope spring bean.

96)Define @Scope(“request”)?

➔If want to store the form data/front end UI of a web application or distributed app in the spring bean class obj which changes request to request then prefer keeping spring bean class obj in the request scope.

97)Define @Scope(“session”)?

➔Keeps the spring bean class obj ref in the session obj as the session attribute value i.e spring bean class object is specified to one browser s/w of client machine.

98)Define @Scope(“application”)?

➔If u want to store request count, users count of a web application in spring bean class obj then better to put spring bean class obj in application scope.

99)What is the difference between singleton scope and application scope in the web application?

➔-Singleton scope means 1 obj for spring bean per bean id with respect to each IOC container.

-application scope means 1 obj for spring bean per bean id with respect to entire web application.

100)Define @Scope("websocket")?

➔In spring programming, when @Scope("websocket") makes the IOC container to keep the spring bean class obj in the websocket.session that supports full duplex communication.

101)What is the pre-instantiation and eager instantiation of the spring bean?

➔-If the IOC container is creating any spring bean class obj during the startup of the IOC container that is called pre-instantiation or eager instatiation of Spring bean.

-Only singleton scope spring bean support pre-instantiation of spring bean.

102)Why there is pre-instantiated only for singleton scope spring beans?

➔- Pre-instantiation occurs only for singleton scope beans because Spring creates a single instance of the bean and reuses it throughout the application's lifetime. Since the singleton bean is shared across the entire application, Spring instantiates it once during startup to ensure it's available whenever needed.

103)How to disable pre-instantiation on singleton scope spring bean?

➔@Lazy(true) along with @Component or @Bean annotation.

104)What is properties file?

➔The text file that maintains the entries in the form of key=value pairs is called properties file.

105) Define @PropertySource?

➔ The @PropertySource annotation in Spring is used to load properties

from an external file (like .properties) into the Spring Environment so you can use them in your application.

106)What is the difference between @Value and @Autowired annotation?

➔-To inject one spring bean class obj to another spring bean class obj's HAS-A property take the support of @Autowired(for injecting spring bean)

-To inject simple values collected from different places(like properties file,system properties and etc..) to the primitive/String bean properties take the support of @Value annotation.

107)How to configure multiple spring bean cfg files in our spring apps?

➔@PropertySource({"com/nt/commons/Info1.properties","com/nt/commons/Info.properties"})

108)If multiple properties files that are configured with spring app are having same keys and with different values then the @Value annotation having that key injects which value as the final value?

➔The lastly configured properties file (in @PropertySource Anntation) key maintained value will be picked up for injecting the value to the spring bean property.

109)What is I18n?

➔The process of making our app working for different Locales is called I18n.

110)Define ctx.getMessage()?

➔Is given to get message from the properties file based on the given key and given Locale object information.

111)What is the dfference b/w ctx.getBean() method and ctx.getMessage() method?

➔ctx.getBean() method gives the spring bean class obj ref based on the

given bean id where as ctx.getMessage() method reads the message of the given key from the properties file that is activated based on the Locale object is that is given.

112)How to change the dependent spring bean of target spring bean when multiple possible dependent spring beans are there?

➔**Can be done in two ways:-**

a)using spring bean cfg file + alias name for bean id + properties file + @Qualifier().—not a great solution becoz we are entertaining the spring bean cfg file(xml file)

b)using profile concept—Best solution to solve the ambiguity problem with loose coupling.

113)What is the difference between BeanFactory and Applicationcontext container?

➔ BeanFactory container	ApplicationContext container
a)It is the object of a class that im-plements BeanFactory(I) directly or indirectly.eg(XmlBeanFactory)	a)It is the object of a class that imple-ments ApplicationContext(I) directly or indirectly.eg(FileSystemXmlApplication Context, AnnotationConfigApplication Context)
b)It is the subset of ApplicationCo-ntext Container.	b)It is the super set of BeanFactory Co-ntainer.
c)Does no support pre-instantiation of singleton scope spring beans.	c)Support for pre-instantiation of single ton scope spring beans.
d)There is no support for proper-ties file and place holder $<key> directly.	d)There is a direct support for propertie file and place holder $(<key>)
e)No support for Internationaliza-tion(118n)	e)Support for Internationalization(118n)
f)Does not support the event hand ling.	f)Support for event handling.
g)Does not give direct support for annotation driven cfgs and 100% code driven cfgs.	g)Direct support for annotation driven cfgs and 100% code driven cfgs.
h)Allow only xml file as the spring	h)Allows xml as the spring bean cfg file

bean cfg file.	and @Configuration class as the configuration class.
i)Can no think about using Spring boot programming.	i)Required Spring boot programming.
j)Light weight container	j)Bit Heavy weight container.

114)Where should i use BeanFactory IOC container and where should I use ApplicationContext container?

➔If spring is used in Memory Sensitive apps like mobile apps, embedded system apps, and IOT apps….where 1 or 2 few extra kilo bytes also matters then prefer using BeanFactory IOC container.

➔If spring is used in web application, distributed Apps, enterprize Apps, and etc…Where Memory utilization does not matter then prefer using ApplicationContext container.

115)For IOC container of 100% Code driven cfgs, if we can not give certain inputs or instruction through java code and annotation then how to pass such special instruction to IOC container?

➔ We can give these special instruction to IOC container through xml file by linking the file with @Configuration class using @importResource annotation.

116)Define @importResource?

➔ The @ImportResource annotation in Spring is used to import XML-based Spring configuration files into a Java-based configuration class.

117)What is the difference between JavaBean and Spring Bean?

➔Java Bean is a java class that is having setter and getters methods. It is always used as the helper class to pass multiple values in the form of single object from one main class to another main class of the same project or different projects.Java Bean are always used as helper classes in the project as serious data carriers.

➔Spring Bean is a java class whose object is created and managed by IOC container, Spring bean class container b.methods having b.logic. spring beans are used as the main comps of the Project taking main logic like b.logics, persistence logic etc.

118)Different approaches of spring bean life cycle programming.

➔**a)Declarative approach(xml cfgs):-**Here life cycle method names are Custom names.use init-method, destroy-method attribute of <bean>tag.

b)Programatic Approach(using interfaces implementation):-Here life cycle method names are fixed name becoz they are taken from the implemented spring api interfaces.(InitliazingBean(I),DisponsibleBean(I)

c)Annotation cfgs Approach(using jdk supplied annotation like @PostConstruct and @PreDestroy)

-Here the life cycle method names are custom names having annotations.

-**@PostConstruct** to configure the java method as the init life cycle.

-@PreDestroy to configure the java method as the init life cycle.

119)What is Java Config Annotation?

➔The annotations that are given by jdk APIs, JEE APIs, and implemented in different frameworks and technologies by providing the functionalities are called Java Config Annotation.

120)Different ways of solving the ambiguity problem related to dependency Injection?

➔a)Using @Primary:-Performs byType mode of Autowiring.

b)Using @Qualifier:-Performs byName mode of Autowiring.

c)By matching target class HAS-A property name with Dependent spring bean class id(By Name mode of Autowiring)

THANK YOU

.