Spring Descriptive Questions

Introduction

1. What is Spring?

Spring is a lightweight, loosely coupled and integrated framework for developing enterprise applications in java.

2. What are the advantages of spring framework?

1. Predefined Templates
2. Loose Coupling
3. Easy to test
4. Lightweight
5. Fast Development
6. Powerful Abstraction
7. Declarative support

3. What are features of Spring?

- Lightweight
- Inversion of control (IOC)
- Aspect oriented (AOP)
- Container
- MVC Framework
- Transaction Management
- JDBC Exception Handling

4. What are the modules of spring framework?

Spring comprises of seven modules. They are...

1. The core container
2. Spring context
3. Spring AOP
4. Spring DAO
5. Spring ORM
6. Spring Web module
7. Spring MVC framework

5. What is Maven Repository?

In Maven terminology, a repository is a directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.
Maven repository are of three types. The following illustration will give an idea regarding these three types.

- local
- central
- remote

Configuration

6. What is autowiring in spring? What are the autowiring modes?

Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic.
Let's see the code to inject bean using dependency injection.

<bean id="emp" class="com.example.Employee" autowire="byName" />

The autowiring modes are given below:

No.	Mode	Description
1)	no	this is the default mode, it means autowiring is not enabled.
2)	byName	injects the bean based on the property name. It uses setter method.
3)	byType	injects the bean based on the property type. It uses setter method.
4)	constructor	It injects the bean using constructor

The "autoDetect" mode is deprecated since spring 3.

7. What do you mean by Bean wiring?

The act of creating associations between application components (beans) within the Spring container is referred to as Bean wiring.

8. What is Spring Java-Based Configuration? Give some annotation example.

Java based configuration option enables you to write most of your Spring configuration without XML but with the help of few Java-based annotations.
An example is the @Configuration annotation, that indicates that the class can be used by the Spring IoC container as a source of bean definitions. Another example is the @Bean annotated method that will return an object that should be registered as a bean in the Spring application context.

9. What is Annotation-based container configuration?

An alternative to XML setups is provided by annotation-based configuration which relies on the **Bytecode** metadata for wiring up components instead of angle-bracket declarations. Instead of using XML to describe a bean wiring, the developer moves the configuration into the component class itself by using annotations on the relevant class, method, or field declaration.

10. How do you turn on annotation wiring?

Annotation wiring is not turned on in the Spring container by default. In order to use annotation based wiring we must enable it in our Spring configuration file by configuring **<context:annotation-config/>** element.

11. @Required annotation

This annotation simply indicates that the affected bean property must be populated at configuration time, through an explicit property value in a bean definition or through autowiring. The container throws BeanInitializationException if the affected bean property has not been populated.

12. @Autowired annotation

The @Autowired annotation provides more fine-grained control over where and how autowiring should be accomplished. It can be used to autowire bean on the setter method just like @Required annotation, on the constructor, on a property or pn methods with arbitrary names and/or multiple arguments.

13. @Qualifier annotation

When there are more than one beans of the same type and only one is needed to be wired with a property, the @Qualifier annotation is used along with @Autowired annotation to remove the confusion by specifying which exact bean will be wired.

14. What are the different bean scopes in spring?

There are 5 bean scopes in spring framework.

No.	Scope	Description
1)	singleton	The bean instance will be only once and same instance will be returned by the IOC container. It is the default scope.
2)	prototype	The bean instance will be created each time when requested.
3)	request	The bean instance will be created per HTTP request.
4)	session	The bean instance will be created per HTTP session.
5)	globalsession	The bean instance will be created per HTTP global session. It can be used in portlet context only.

IOC And DI

15. What is IOC and DI?

IOC (Inversion of Control) and DI (Dependency Injection) is a design pattern to provide loose coupling. It removes the dependency from the program.

Let's write a code without following IOC and DI.

```
public class Employee{
        Address address;
        Employee(){
            address=new Address();  //creating instance
                }
        }
```

Now, there is dependency between Employee and Address because Employee is forced to use the same address instance.

Let's write the IOC or DI code.

```
public class Employee{
        Address address;
        Employee(Address address){
                this.address=address;  //not creating instance
                        }
        }
```

Now, there is no dependency between Employee and Address because Employee is not forced to use the same address instance. It can use any address instance.

16. What are the different types of IOC (dependency injection)?

There are three types of dependency injection:

- **Constructor Injection**: Dependencies are provided as constructor parameters.
- **Setter Injection**: Dependencies are assigned through JavaBeans properties (ex: setter methods).
- **Interface Injection**: Injection is done through an interface.

17. What are the benefits of IOC (Dependency Injection)?

Benefits of IOC (Dependency Injection) are as follows:

- Minimizes the amount of code.
- Make application more testable
- Loose coupling
- IOC containers support eager instantiation and lazy loading of services

18. What is the role of IOC container in spring?

IOC container is responsible to:

- create the instance
- configure the instance, and
- assemble the dependencies

19. What are the types of IOC container in spring?

There are two types of IOC containers in spring framework.

1. BeanFactory
2. ApplicationContext

20. What is the difference between BeanFactory and ApplicationContext?

a. BeanFactory is the **basic container** whereas ApplicationContext is the **advanced container**.
b. ApplicationContext extends the BeanFactory interface.
c. ApplicationContext provides more facilities than BeanFactory such as integration with spring AOP, message resource handling for i18n, load file resources, ResourceLoader support etc.

21. What is the difference between constructor injection and setter injection?

No.	Constructor Injection	Setter Injection
1)	No Partial Injection	Partial Injection
2)	Doesn’t override the setter property	Overrides the constructor property if both are defined.
3)	Creates new instance if any modification occurs	Doesn't create new instance if you change the property value
4)	Better for too many properties	Better for few properties.

22. What is BeanFactory?

A BeanFactory is like a factory class that contains a collection of beans. The BeanFactory holds Bean Definitions of multiple beans within itself and then instantiates the bean whenever asked for by clients.

23. What is Application Context?

A bean factory is fine to simple applications, but to take advantage of the full power of the Spring framework, you may want to move up to Springs more advanced container, the application context. On the surface, an application context is same as a bean factory.Both load bean definitions, wire beans together, and dispense beans upon request. But it also provides:

- A means for resolving text messages, including support for internationalization.
- A generic way to load file resources.
- Events to beans that are registered as listeners.

24. What are the common implementations of the Application Context?

The three commonly used implementation of 'Application Context' are

I. **ClassPathXmlApplicationContext:** It Loads context definition from an XML file located in the classpath, treating context definitions as classpath resources. The application context is loaded from the application's classpath by using the code.
ApplicationContext context = new ClassPathXmlApplicationContext("bean.xml");

II. **FileSystemXmlApplicationContext:** It loads context definition from an XML file in the filesystem. The application context is loaded from the file system by using the code.
ApplicationContext context = new FileSystemXmlApplicationContext("bean.xml");

III. **XmlWebApplicationContext:** It loads context definition from an XML file contained within a web application.

25. What is the typical Bean life cycle in Spring Bean Factory Container?

Bean life cycle in Spring Bean Factory Container is as follows:

- The spring container finds the bean's definition from the XML file and instantiates the bean.
- Using the dependency injection, spring populates all of the properties as specified in the bean definition
- If the bean implements the BeanNameAware interface, the factory calls setBeanName() passing the bean's ID.
- If the bean implements the BeanFactoryAware interface, the factory calls setBeanFactory(), passing an instance of itself.
- If there are any BeanPostProcessors associated with the bean, their post-ProcessBeforeInitialization() methods will be called.
- If an init-method is specified for the bean, it will be called.
- Finally, if there are any BeanPostProcessors associated with the bean, their postProcessAfterInitialization() methods will be called.

26. What is the difference Between IoC and DI?

IoC is a generic term meaning rather than having the application, call the methods in a framework, the framework calls implementations provided by the application.
DI is a form of IoC, where implementations are passed into an object through constructors/setters/service lookups, which the object will 'depend' on in order to behave correctly.

AOP

27. What is AOP?

AOP or Aspect Oriented Programming is a methodology that divides the program logic into pieces or parts or concerns. It increases the modularity and the key unit is Aspect.

28. What are the advantages of spring AOP?

AOP enables you to dynamically add or remove concern before or after the business logic. It is **pluggable** and **easy to maintain**.

29. How the AOP used in Spring?

To provide declarative enterprise services, especially as a replacement for EJB declarative services. The most important such service is declarative transaction management, which builds on the Spring Framework's transaction abstraction. To allow users to implement custom aspects, complementing their use of OOP with AOP.

30. What are the AOP terminology?

AOP terminologies or concepts are as follows:

- JoinPoint
- Advice
- Pointcut
- Aspect
- Introduction
- Target Object
- Interceptor
- AOP Proxy
- Weaving

31. What is JoinPoint?

JoinPoint is any point in program such as field access, method execution, exception handling etc. In Spring AOP, a join point always represents a method execution.

32. Does spring framework support all JoinPoints?

No, spring framework supports method execution joinpoint only.

33. What is Advice?

Advice represents action taken by an aspect at a particular JoinPoint. Different types of advice include "around," "before" and "after" advice. Many AOP frameworks, including Spring, model an advice as an interceptor, maintaining a chain of interceptors "around" the join point.

34. What are the types of advice in AOP?

There are 5 types of advices in spring AOP.

1. Before Advice
2. After Advice
3. After Returning Advice
4. Throws Advice
5. Around Advice

35. What is Pointcut?

Pointcut is expression language of Spring AOP.

36. What is Aspect?

Aspect is a class in spring AOP that contains advices and JoinPoints. In Spring AOP, aspects are implemented using regular classes (the schema-based approach) or regular classes annotated with the @Aspect annotation (@AspectJ style).

37. What is Introduction?

Introduction represents introduction of new fields and methods for a type.

38. What is target object?

Target Object is a proxy object that is advised by one or more aspects.

39. What is interceptor?

Interceptor is a class like aspect that contains one advice only.

40. What is weaving?

Weaving is a process of linking aspect with other application.

41. Does spring perform weaving at compile time?

No, spring framework performs weaving at runtime.

42. What are the AOP implementation?

There are 3 AOP implementation.

1. Spring AOP
2. Apache AspectJ
3. JBoss AOP

JDBC

43. What is database?

A database is a collection of data. A database is basically a collection of information organized in such a way that a computer program can quickly select desired pieces of data. Traditional databases are organized by fields, records, and files. A field is a single piece of information; a record is one complete set of fields; and a file is a collection of records.

44. Spring DAO support

The Data Access Object (DAO) support in Spring is aimed at making it easy to work with data access technologies like JDBC, Hibernate or JDO in a consistent way. This allows us to switch between the persistence technologies fairly easily and to code without worrying about catching exceptions that are specific to each technology.

45. What is the Main objective of an ORM library?

The main objective of an ORM library is to close the gap between the relational data structure in the RDBMS and the OO model in Java so that developers can focus on programming with the object model and at the same time easily perform actions related to persistence.

46. What is JdbcTemplate Class?

Jdbc class is a class which represents the core of Spring's JDBC support. It can execute all types of SQL statements. In the most simplistic view, you can classify the data definition and data manipulation statements.

47. What is JdbcTemplate?

The JdbcTemplate is the spring answer to plain JDBC access. It handles database connectivity. Users can specify the sql and a callback through its method. The template does the work of creating the connection and executing the query or update. The callback can be used to handle resultset (ResultSetExtractor) or create a prepared statement (PreparedStatementCreator)

48. What are the advantages of JdbcTemplate in spring?

Less code: By using the JdbcTemplate class, you don't need to create connection, statement, start transaction, commit transaction and close connection to execute different queries. We can execute the query directly. This allows the programmer to react more flexible to the errors. The Spring JDBC template converts also the vendor specific error messages into better understandable error messages.

49. What are classes for spring JDBC API?

1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate
4. SimpleJdbcInsert
5. SimpleJdbcCall

50. What is the advantage of NamedParameterJdbcTemplate?

NamedParameterJdbcTemplate class is used to pass value to the named parameter. A named parameter is better than? (question mark of PreparedStatement).
It is **better to remember**.

51. What is the advantage of SimpleJdbcTemplate?

The **SimpleJdbcTemplate** supports the feature of var-args and autoboxing.

52. How can you fetch records by spring JdbcTemplate?

You can fetch records from the database by the **query method of JdbcTemplate**. There are two interfaces to do this:

1. ResultSetExtractor
2. RowMapper

53. What is Spring's RowMapper<T>?

Spring's RowMapper<T> interface provides a simple way to perform mapping from a JDBC resultset to POJO's.

54. What is the difference of datasource and database connection?

DataSource provides and manages Connections.

55. Write down some JDBC package name in spring?

- org.springframework.jdbc.core
- org.springframework.jdbc.datasource
- org.springframework.jdbc.object
- org.springframework.jdbc.support
- org.springframework.jdbc.config

56. What are embedded database support?

The **org.springframework.jdbc.datasource.embedded** package provides support for embedded Java database engines. Support for HSQL, H2, and Derby is provided natively. You can also use an extensible API to plug in new embedded database types and DataSource implementations.

57. ORM's Spring support

Spring supports the following ORM's:

- Hibernate

- iBatis
- JPA (Java Persistence API)
- TopLink
- JDO (Java Data Objects)
- OJB

58. What are the ERD, ORM?

ERD: Entity–relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database.
ORM: Object-relational mapping (ORM) is a programming technique in which a metadata descriptor is used to connect object code to a relational database. Object code is written in object-oriented programming (OOP) languages such as Java. ORM converts data between type systems that are unable to coexist within relational databases and OOP languages.

59. How can JDBC be used more efficiently in the Spring framework?

When using the Spring JDBC framework the burden of resource management and error handling is reduced. So developers only need to write the statements and queries to get the data to and from the database. JDBC can be used more efficiently with the help of a template class provided by Spring framework, which is the JdbcTemplate

60. What is the function of RowMapper?

RowMapper Interface is used by the JdbcTemplate to map a resultset row. The implementation of this interface maps a resultset row to a result object. The implementation does not have to worry about catching exceptions. Implementations must implement the method

Hibernate

61. How do you access Hibernate using spring?

There are two ways to Spring's Hibernate integration:

a. By Inversion of Control with a Hibernate Template and Callback
b. By extending Hibernate Dao Support and Applying an AOP Interceptor

62. How would you integrate Spring and Hibernate using Hibernate Dao Support?

This can be done through Spring's SessionFactory called LocalSessionFactory. The steps in integration process are:

a. Configure the Hibernate SessionFactory
b. Extend your DAO Implementation from Hibernate Dao Support
c. Wire in Transaction Support with AOP

63. Write the Hibernate dependency code?

```
<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-entitymanager</artifactId>
        <version>3.6.0.Final</version>
    </dependency>
</dependencies>
```

64. Write down the mandatory hibernate configurations.

- The dataSource bean

- The transactionManager bean
- Component scan
- Hibernate SessionFactory bean

65. How many mapping type have in hibernate?

a) One-to-Many Mappings
b) Many-to-Many Mappings

66. Why we use hibernate?

We can simply integrate hibernate application with spring application.

- Hibernate generates very efficient queries very consistently.
- Hibernate spares you an unnecessary database calls. So all these lookup tables and rarely changing data can get cached and much more.
- Hibernates portability across the relational databases is amazing.

Data Access

67. Why we use JPA2?

Like many other Java specification requests, the objective of the JPA2 specification is to standardize the ORM programming model in both the JSE and JEE environments. It defines a common set of concepts, annotations, interfaces and other services that a JPA persistence provider should implement.

68. What Spring Data JPA?

The Spring Data JPA project is a subproject under the Spring Data umbrella project. The main objective of the Spring Data JPA project is to provide additional features for simplifying application development with JPA.

69. The main configurations of JPA2

- The dataSource bean
- The transactionManager bean
- Component scan
- JPA EntityManagerFactory bean (emf)

70. What is Data Access Layer?

Data Access Layer provides to our application components with a standard mechanism for storing and retrieving data.

Design And Implementation

71. Why do we need the concept of Design to Interfaces?

a. Easy to swap out a component if it becomes problematic
b. Gained by this loose coupling is the increase in testability
c. We can swap out an implementation of an interface for a mock implementation, which allows us more flexibility when testing.

72. What is DOM and goals of DOM?

A Domain Object Model (DOM) is a set of classes that model concepts from the problem domain. It is an object-based representation of the application problem domain, intended to allow the programmer to code in terms of objects that exist in the problem domain.

Goal of DOM-

- How the problem domain is structured?
- How the domain objects will be used?
- How the underlying data store is constructed?

73. What is Interface-driven design?

Interface-driven design is a traditional OOP best practice, In Interface-driven design the main components of the application are defined in terms of interfaces rather than concrete classes.

74. What are the drawbacks of the Basic factory pattern?

The factory pattern has three main drawbacks; they are:

- There is no way to change an implementing class without a recompile.
- There is no way to make multiple implementations available transparently to different components.
- There is no way simply to switch instantiation models.

75. What is service layer?

A service layer acts as a sort of gateway into application, provident presentation code with a simple, unified way to get at business logic or a good service layer also serves as a definition of what application can actually perform and what logic is available to be presented to the user. (When one class provide service to another class that is called service layer).

76. Difference between domain objects and value objects?

A DOM is an object-based representation of the application problem domain, while a value object purely encompasses state. It is perfectly acceptable for a domain object to encompass both state and behavior.

Another key difference between domain objects and value objects is that a value object's structure is driven by the need to transfer data remotely, whereas a domain object is modeled to represent a real world concept and is not driven by some need of the application infrastructure.

77. In which scenario, you will use singleton and prototype scope?

Singleton scope should be used with EJB **stateless session bean** and prototype scope with EJB **stateful session bean**.

78. What is Factory pattern in spring?

The Factory Pattern defines a class whose responsibility is to provide application components with implementations of other application components.

Transaction Management

79. What are the benefits of the Spring Framework's transaction management?

- It provides a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO.

- It provides a simpler API for programmatic transaction management than a number of complex transaction APIs such as JTA.
- It supports declarative transaction management.
- It integrates very well with Spring's various data access abstractions.

80. What are the transaction management supports provided by spring?

Spring framework provides two type of transaction management supports:

1. **Programmatic Transaction Management**: should be used for few transaction operations.
2. **Declarative Transaction Management**: should be used for many transaction operations.

81. Which Transaction management type is more preferable?

Most users of the Spring Framework choose declarative transaction management because it is the option with the least impact on application code and hence is most consistent with the ideals of a non-invasive lightweight container. Declarative transaction management is preferable over programmatic transaction management though it is less flexible than programmatic transaction management, which allows you to control transactions through your code.

Type Conversion And Validation

82. What is Spring Type Conversion system?

In Spring 3, a new type conversion system was introduced, providing a powerful way to convert between any Java types within Spring-powered applications. It can perform the same functionality provided by the **PropertyEditor** support.

83. How is using Spring Validator Interface?

Using Spring's Validator interface, we can develop some validation logic by creating a class to implement the Validator interface and implements two methods. The **supports()** method indicates whether validation of the passed-in class type is supported by the validator. The **validate()** method performs validation on the passed-in object. The result will be stored in an instance of the **org.springframework.validation.Errors** interface.

84. What is Validation in Spring?

Validation is a critical part of any application. Validation rules applied on domain objects ensure that all business data is well structured and fulfills all the business definitions. Using Spring's Validator interface, we can develop some validation logic by creating a class to implement the interface.

85. Which is Validation API to Use?

JSR-303 Bean Validation.

86. Defining Validation Constraints on Object Properties?

@NotNull annotation, @Size annotation etc.

MVC

87. What do mean by Spring MVC?

In the spring framework, the Spring MVC module provides comprehensive support for the MVC pattern with support for other feature (for example theming, i18n, validation, type conversion, formatting and so on) that ease the implementation of the presentation

88. What is the front controller class of Spring MVC?

The **DispatcherServlet** class works as the front controller in Spring MVC.

89. What does @Controller annotation?

The **@Controller** annotation marks the class as controller class. It is applied on the class.

90. What is the function of @Controller annotation?

The @Controller annotation indicates that a particular class serves the role of a controller. Spring does not require you to extend any controller base class or reference the Servlet API.

91. What does @RequestMapping annotation?

The **@RequestMapping** annotation maps the request with the method. It is applied on the method.

92. What is the function of @RequestMapping annotation?

@RequestMapping annotation is used to map a URL to either an entire class or a particular handler method.

93. What does the ViewResolver class?

The **ViewResolver** class resolves the view component to be invoked for the request. It defines prefix and suffix properties to resolve the view component.

94. Which ViewResolver class is widely used?

The **org.springframework.web.servlet.view.InternalResourceViewResolver** class is widely used.

95. Does spring MVC provide validation support?

Yes.

96. What are the major considerations when developing web applications?

- Performance
- User-friendly
- Interactive and richness
- Accessibility

97. Detail of MVC pattern?

Model: A model represents the business data as well as the “state” of the application within the context of the user.
View: This presents the data to the user in the desired format, supports interaction with users, and supports client-side validation, i18n, styles, and so on.
Controller: The controller handles requests for actions performed by users in the frontend, interacting with the service layer, updating the model and directing users to the appropriate view based on the result of execution.

98. The main components of Spring MVC request life cycle?

- Filter
- Dispatcher servlet
- Common services
- Handler mapping
- Handler interceptor
- Handler exception resolver
- View Resolver

99. What is the function of DispatcherServlet?

The Spring Web MVC framework is designed around a DispatcherServlet that handles all the HTTP requests and responses.

100. What is WebApplicationContext?

The WebApplicationContext is an extension of the plain **ApplicationContext** that has some extra features necessary for web applications. It differs from a normal **ApplicationContext** in that it is capable of resolving themes and that it knows which servlet it is associated with.

101. What is Controller in Spring MVC framework?

Controllers provide access to the application behavior that you typically define through a service interface. Controllers interpret user input and transform it into a model that is represented to the user by the view. Spring implements a controller in a very abstract way, which enables you to create a wide variety of controllers.

102. How to configure Spring MVC support for web applications?

We need to perform the following configurations in the web deployment descriptor:

- Configuring the root WebApplicationContext
- Configuring the servlet filters required by Spring MVC
- Configuring the dispatcher servlets within the application

Web Flow And JSF

103. What is Spring Web Flow?

Spring Web Flow is a framework for building flow-based applications. Spring web flow is a limited page flow functionality offered by classic MVC frameworks. Web Flow integrates nicely with Spring MVC, JSF or Struts and offers the means to define page flows consisting of views and actions and in addition allows page flow reuse.

104. What are the modules of spring web flow?

- ✓ Spring-web-flow
- ✓ Spring-faces
- ✓ Spring JavaScript
- ✓ Spring Binding

105. Describe the main concept of Spring web flow.
Or Write down the Spring Web Flow Features.

- **Flow**: A flow consists of a series of steps called sates.
- **View**: The same as in the MVC pattern, a view is a user interfaces that presents the state of the model to the user and provide the user interaction.

- **Conversation**: In a web application, in terms of bean scopes, there are three type namely request, session and application.

106. Write some Bean Scope names in Spring Web Flow?

- ✓ Flow
- ✓ View
- ✓ Request
- ✓ Flash
- ✓ Conversation

107. JSF Application Life Cycle

- Restore view
- Apply request
- Process validations
- Update model values
- Invoke application
- Render response

108. What is JSF?

JSF is a request driven MVC web framework based on a component driven UI design model.