1. What is Spring?

Spring is an open source development framework for [Enterprise Java](https://www.javacodegeeks.com/tutorials/java-tutorials/enterprise-java-tutorials/). The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make Java EE development easier to use and promote good programming practice by enabling a [POJO-based programming model](https://www.javacodegeeks.com/2012/09/how-to-write-better-pojo-services.html).

2. What are benefits of Spring Framework?

* **Lightweight:**Spring is lightweight when it comes to size and transparency. The basic version of spring framework is around 2MB.
* **Inversion of control (IOC):** Loose coupling is achieved in Spring, with the [Inversion of Control technique](https://www.javacodegeeks.com/2011/08/what-is-dependency-inversion-is-it-ioc.html). The objects give their dependencies instead of creating or looking for dependent objects.
* **Aspect oriented (AOP):** [Spring supports Aspect oriented programming](https://www.javacodegeeks.com/2011/01/aspect-oriented-programming-spring-aop.html) and separates application business logic from system services.
* **Container:** Spring contains and manages the life cycle and configuration of application objects.
* **MVC Framework:** Spring’s web framework is a well-designed [web MVC framework](https://www.javacodegeeks.com/2011/02/spring-mvc-development-tutorial.html), which provides a great alternative to web frameworks.
* **Transaction Management:** Spring provides a consistent transaction management interface that can scale down to a local transaction and scale up to global transactions (JTA).
* **Exception Handling:** Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO) into consistent, unchecked exceptions.

3. Which are the Spring framework modules?

There are around 20 modules which are generalized into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test. The basic modules of the Spring framework are :

Spring Core Container

This layer is basically the core of Spring Framework. It contains the following modules:

* Core module
* Bean module
* Context module
* [Expression Language module](https://examples.javacodegeeks.com/enterprise-java/spring/spel/spring-expression-language-example/)

[Data Access/Integration](https://www.javacodegeeks.com/spring-data-tutorials)

This layer provides support to interact with the database. It contains the following modules:

* [JDBC module](https://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/)
* [Object-Relational Mapping (ORM) module](http://examples.javacodegeeks.com/enterprise-java/spring/jpaorm/spring-hibernate-mysql-and-maven-showcase/)
* [Java Messaging Service (JMS) module](https://www.javacodegeeks.com/2015/09/enterprise-messaging.html)
* Object XML Mappers (OXM) module
* [Transaction Management module](https://examples.javacodegeeks.com/core-java/sql/jdbc-transaction-management-example/)

Web

This layer provides support to create web application. It contains the following modules:

* Web module
* [Web-MVC module](https://www.javacodegeeks.com/spring-mvc-tutorials)
* Web-Socket module
* Web-Portlet module

Aspect Oriented Programming ([AOP](https://examples.javacodegeeks.com/enterprise-java/spring/aop/spring-aop-example/))

In this layer you can use Advices, Pointcuts etc., to decouple the code.  
Instrumentation – This layer provides support to class instrumentation and classloader implementations.

Test

This layer provides support to testing with JUnit and TestNG.

Messaging

This module provides support for STOMP. It also supports an annotation programming model that is used for routing and processing STOMP messages from WebSocket clients.

Aspects

This module provides support to integration with [AspectJ](https://examples.javacodegeeks.com/enterprise-java/spring/aop/spring-aop-aspectj-example/).

4. Explain the Core Container (Application context) module

This is the basic Spring module, which provides the fundamental functionality of the Spring framework. BeanFactory is the heart of any spring-based application. Spring framework was built on the top of this module, which makes the Spring container.

5. BeanFactory implementation example

A BeanFactory is an implementation of the factory pattern that applies Inversion of Control to separate the application’s configuration and dependencies from the actual application code.

The most commonly used BeanFactory implementation is the XmlBeanFactory class.

6. XMLBeanFactory

The most useful one is org.springframework.beans.factory.xml.XmlBeanFactory, which loads its beans based on the definitions contained in an XML file. This container reads the configuration metadata from an XML file and uses it to create a fully configured system or application.

7. Explain the AOP module

The AOP module is used for developing aspects for our Spring-enabled application. Much of the support has been provided by the AOP Alliance in order to ensure the interoperability between Spring and other AOP frameworks. This module also introduces metadata programming to Spring.

8. Explain the JDBC abstraction and DAO module

With the [JDBC abstraction and DAO module](http://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/) we can be sure that we keep up the database code clean and simple, and prevent problems that result from a failure to close database resources. It provides a layer of meaningful exceptions on top of the error messages given by several database servers. It also makes use of Spring’s AOP module to provide transaction management services for objects in a Spring application.

9. Explain the object/relational mapping integration module

Spring also supports for using of an object/relational mapping (ORM) tool over straight JDBC by providing the ORM module. Spring provides support to tie into several popular ORM frameworks, including Hibernate, JDO, and iBATIS SQL Maps. Spring’s transaction management supports each of these ORM frameworks as well as JDBC.

10. Explain the web module

The Spring web module is built on the application context module, providing a context that is appropriate for web-based applications. This module also contains support for several web-oriented tasks such as transparently handling multipart requests for file uploads and programmatic binding of request parameters to your business objects. It also contains integration support with Jakarta Struts.

11. Explain the Spring MVC module

MVC framework is provided by Spring for building web applications. Spring can easily be integrated with other MVC frameworks, but Spring’s MVC framework is a better choice, since it uses IoC to provide for a clean separation of controller logic from business objects. With Spring MVC you can declaratively bind request parameters to your business objects.

12. Spring configuration file

Spring configuration file is an XML file. This file contains the classes information and describes how these classes are configured and introduced to each other.

13. How can we have multiple Spring configuration files?

* web.xml contextConfigLocation: you can load them all into your Web application context via the ContextConfigLocation element. You’re already going to have your primary applicationContext here, assuming you’re writing a web application. All you need to do is put some white space between the declaration of the next context.
* applicationContext.xml import resource: you can add your primary applicationContext.xml to the web.xml and then use import statements in that primary context.

14. What are the common implementations of the ApplicationContext?

* The **FileSystemXmlApplicationContext** container loads the definitions of the beans from an XML file. The full path of the XML bean configuration file must be provided to the constructor.
* The **ClassPathXmlApplicationContext** container also loads the definitions of the beans from an XML file. Here, you need to set CLASSPATH properly because this container will look bean configuration XML file in CLASSPATH.
* The **WebXmlApplicationContext:** container loads the XML file with definitions of all beans from within a web application.

15. What is the difference between Bean Factory and ApplicationContext?

Application contexts provide a means for resolving text messages, a generic way to load file resources (such as images), they can publish events to beans that are registered as listeners. In addition, operations on the container or beans in the container, which have to be handled in a programmatic fashion with a bean factory, can be handled declaratively in an application context. The application context implements MessageSource, an interface used to obtain localized messages, with the actual implementation being pluggable.

16. What are some of the best practices for Spring Framework?

Some of the [best practices for Spring Framework](https://examples.javacodegeeks.com/enterprise-java/spring/20-spring-framework-best-practices/) are:

* Define singleton beans with names same as their class or interface names
* Place Spring bean configuration files under a folder instead of root folder
* Give common prefixes or suffixes to Spring bean configuration files
* Avoid using import elements within Spring XML configuration files as much as possible
* Stay away from auto wiring in XML based bean configurations
* Always externalize bean property values with property placeholders
* Select default version-less XSD when importing namespace definitions
* Always place classpath prefix in resource paths
* Create a setter method even though you use field level auto wiring
* Create a separate service layer even though service methods barely delegate their responsibilities to corresponding DAO methods

17. What are the various ways of using Spring Framework?

You can use Spring Framework:

* for writing web applications
* for exposing RESTful services
* to secure your web applications
* for communicating with databases
* for handling long running jobs
* to handle external resources or systems you have to work with
* for testing purposes
* for standalone java projects
* to convert your application into an executable
* to integrate Social Media into your applications

18. How can we use Spring to create Restful Web Service returning JSON response?

Any Spring @RestController in a Spring Boot application should render JSON response by default as long as Jackson2 is on the classpath.

19. Spring vs Spring MVC vs Spring Boot?

* Spring: the most important feature of Spring is Dependency Injection or Inversion of Control.
* Spring MVC: is a complete HTTP oriented MVC framework managed by the Spring Framework and based in Servlets. It would be equivalent to JSF in the JavaEE stack.
* Spring Boot: is a utility for setting up applications quickly, offering an out of the box configuration in order to build Spring powered applications.

20. What does a Spring application look like?

* Interface: An interface that defines the functions.
* Bean class: It contains properties, its setter and getter methods, functions etc.
* Spring AOP: Provides the functionality of cross-cutting concerns.
* The configuration XML file: Contains the information of classes and how to configure them.
* The Client program: uses the function.

B.Dependency Injection

21. What is Spring IoC container?

The Spring IoC is responsible for creating the objects,managing them with dependency injection (DI), wiring them together, configuring them, as also managing their complete lifecycle.

22. What are the benefits of IOC?

* IOC or dependency injection minimizes the amount of code in an application.
* It makes easy to test applications, since no singletons or JNDI lookup mechanisms are required in unit tests.
* Loose coupling is promoted with minimal effort and least intrusive mechanism.
* IOC containers support eager instantiation and lazy loading of services.

23. How many types of IOC containers are there in spring?

* BeanFactory: A BeanFactory is essentially nothing more than the interface for an advanced factory capable of maintaining a registry of different beans and their dependencies. The BeanFactory enables you to read bean definitions and access them using the bean factory.
* ApplicationContext: The ApplicationContext is the central interface within a Spring application for providing configuration information to the application. It is read-only at run time, but can be reloaded if necessary and supported by the application. A number of classes implement the ApplicationContext interface, allowing for a variety of configuration options and types of applications.

24. BeanFactory vs ApplicationContext

Application Context:

* Bean instantiation/wiring
* Automatic BeanPostProcessor registration
* Automatic BeanFactoryPostProcessor registration
* Convenient MessageSource access (for i18n)
* ApplicationEvent publication

BeanFactor:

* Bean instantiation/wiring

25. What is Dependency Injection in Spring?

Dependency Injection, an aspect of Inversion of Control (IoC), is a general concept, and it can be expressed in many different ways.This concept says that you do not create your objects but describe how they should be created. You don’t directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container (the IOC container) is then responsible for hooking it all up.

26. What is the difference between Tight Coupling and Loose Coupling?

Tight Coupling:

* Tight coupling is when a group of classes are highly dependent on one another.

Loose Coupling:

* Loose coupling is achieved by means of a design that promotes single-responsibility and separation of concerns.

27. What are the different types of IoC (dependency injection)?

* **Constructor-based dependency injection:** Constructor-based DI is accomplished when the container invokes a class constructor with a number of arguments, each representing a dependency on other class.
* **Setter-based dependency injection:** Setter-based DI is accomplished by the container calling setter methods on your beans after invoking a no-argument constructor or no-argument static factory method to instantiate your bean.

28. Which DI would you suggest Constructor-based or setter-based DI?

You can use both Constructor-based and Setter-based Dependency Injection. The best solution is using constructor arguments for mandatory dependencies and setters for optional dependencies.

C.Spring Beans

29. What are Spring beans?

The [Spring Beans](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-3-bean-reference-example/) are Java Objects that form the backbone of a Spring application. They are instantiated, assembled, and managed by the Spring IoC container. These beans are created with the configuration metadata that is supplied to the container, for example, in the form of XML <bean/> definitions.

Beans defined in spring framework are singleton beans. There is an attribute in bean tag named "singleton" if specified true then bean becomes singleton and if set to false then the bean becomes a prototype bean. By default it is set to true. So, all the beans in spring framework are by default singleton beans.

30. What does a Spring Bean definition contain?

A Spring Bean definition contains all configuration metadata which is needed for the container to know how to create a bean, its lifecycle details and its dependencies.

31. How do you provide configuration metadata to the Spring Container?

There are three important methods to provide configuration metadata to the Spring Container:

* XML based configuration file.
* Annotation-based configuration
* [Java-based configuration](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-3-java-config-example/)

32. How do you define the scope of a bean?

When defining a <bean> in Spring, we can also declare a scope for the bean. It can be defined through the scope attribute in the bean definition. For example, when Spring has to produce a new bean instance each time one is needed, the bean’s scope attribute to be prototype. On the other hand, when the same instance of a bean must be returned by Spring every time it is needed, the the bean scope attribute must be set to singleton.

33. Explain the bean scopes supported by Spring

There are five scoped provided by the Spring Framework supports following five scopes:

* In **singleton** scope, Spring scopes the bean definition to a single instance per Spring IoC container.
* In **prototype** scope, a single bean definition has any number of object instances.
* In **request** scope, a bean is defined to an HTTP request. This scope is valid only in a web-aware Spring ApplicationContext.
* In **session**scope, a bean definition is scoped to an HTTP session. This scope is also valid only in a web-aware Spring ApplicationContext.
* In **global-session** scope, a bean definition is scoped to a global HTTP session. This is also a case used in a web-aware Spring ApplicationContext.

The default scope of a Spring Bean is Singleton.

34. Are Singleton beans thread safe in Spring Framework?

No, singleton beans are not thread-safe in Spring framework.

35. Explain Bean lifecycle in Spring framework

* The spring container finds the bean’s definition from the XML file and instantiates the bean.
* Spring populates all of the properties as specified in the bean definition (DI).
* If the bean implements BeanNameAware interface, spring passes the bean’s id to setBeanName() method.
* If Bean implements BeanFactoryAware interface, spring passes the beanfactory to setBeanFactory() method.
* If there are any bean BeanPostProcessors associated with the bean, Spring calls postProcesserBeforeInitialization() method.
* If the bean implements IntializingBean, its afterPropertySet() method is called. If the bean has init method declaration, the specified initialization method is called.
* If there are any BeanPostProcessors associated with the bean, their postProcessAfterInitialization() methods will be called.
* If the bean implements DisposableBean, it will call the destroy() method.

36. Which are the important beans lifecycle methods? Can you override them?

There are two important bean lifecycle methods. The first one is setup which is called when the bean is loaded in to the container. The second method is the teardown method which is called when the bean is unloaded from the container.  
The bean tag has two important attributes (init-method and destroy-method) with which you can define your own custom initialization and destroy methods. There are also the correspondive annotations(@PostConstruct and @PreDestroy).

37. What are inner beans in Spring?

When a bean is only used as a property of another bean it can be declared as an inner bean. Spring’s XML-based configuration metadata provides the use of <bean/> element inside the <property/> or <constructor-arg/> elements of a bean definition, in order to define the so-called inner bean. Inner beans are always anonymous and they are always scoped as prototypes.

38. How can you inject a Java Collection in Spring?

Spring offers the following types of [collection configuration elements](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-collections-list-set-map-and-properties-example/):

* The <list> type is used for injecting a list of values, in the case that duplicates are allowed.
* The <set> type is used for wiring a set of values but without any duplicates.
* The <map> type is used to inject a collection of name-value pairs where name and value can be of any type.
* The <props> type can be used to inject a collection of name-value pairs where the name and value are both Strings.

39. What is bean wiring?

Wiring, or else bean wiring is the case when beans are combined together within the Spring container. When wiring beans, the Spring container needs to know what beans are needed and how the container should use dependency injection to tie them together.

40. What is bean autowiring?

The Spring container is able to [autowire relationships](http://examples.javacodegeeks.com/enterprise-java/spring/beans-spring/spring-autowire-example/) between collaborating beans. This means that it is possible to automatically let Spring resolve collaborators (other beans) for a bean by inspecting the contents of the BeanFactory without using <constructor-arg> and <property> elements.

41. Explain different modes of autowiring?

The autowiring functionality has five modes which can be used to instruct Spring container to use autowiring for dependency injection:

* **no:** This is default setting. Explicit bean reference should be used for wiring.
* **byName:** When autowiring byName, the Spring container looks at the properties of the beans on which autowire attribute is set to byName in the XML configuration file. It then tries to match and wire its properties with the beans defined by the same names in the configuration file.
* **byType:** When autowiring by datatype, the Spring container looks at the properties of the beans on which autowire attribute is set to byType in the XML configuration file. It then tries to match and wire a property if its type matches with exactly one of the beans name in configuration file. If more than one such beans exist, a fatal exception is thrown.
* **constructor:** This mode is similar to byType, but type applies to constructor arguments. If there is not exactly one bean of the constructor argument type in the container, a fatal error is raised.
* **autodetect:**Spring first tries to wire using autowire by constructor, if it does not work, Spring tries to autowire by byType.

42. Are there limitations with autowiring?

Limitations of autowiring are:

* **Overriding:**You can still specify dependencies using <constructor-arg> and <property> settings which will always override autowiring.
* **Primitive data types:** You cannot autowire simple properties such as primitives, Strings, and Classes.
* **Confusing nature:** Autowiring is less exact than explicit wiring, so if possible prefer using explicit wiring.

43. Can you inject null and empty string values in Spring?

Yes, you can.

D.Spring Annotations

44. What are some of the important Spring annotations?

Some of the Spring annotations that I have used in my project are:

* @Component is used to indicate that a class is a component. These classes are used for auto-detection and configured as bean when annotation based configurations are used.
* @Controller is a specific type of component, used in MVC applications and mostly used with @RequestMapping annotation.
* @Repository annotation is used to indicate that a component is used as repository and a mechanism to store/retrieve/search data. We can apply this annotation with DAO pattern implementation classes.
* @Service is used to indicate that a class is a Service. Usually, the business facade classes that provide some services are annotated with this.
* @Required – 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.
* @ResponseBody – for sending Object as response, usually for sending XML or JSON data as response.
* @PathVariable – for mapping dynamic values from the URI to handler method arguments.
* @Autowired – 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.
* @Qualifier – 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.
* @Scope – for configuring scope of the spring bean.
* @Configuration – indicates that the class can be used by the Spring IoC container as a source of bean definitions.
* @ComponentScan – all the classes available under a package will be scanned when this annotation is applied.
* @Bean – for java based configurations, tells spring that a method annotated with @Bean will return an object that should be registered as a bean in the spring application context.
* AspectJ annotations for configuring aspects and advices, @Aspect, @Before, @After, @Around, @Pointcut etc.

45. What does the @RequestParam annotation do?

The [@RequestParam annotation](https://examples.javacodegeeks.com/enterprise-java/spring/spring-requestparam-annotation-example/) in spring binds the parameter values of a query string to the method argument of a controller.

**46. What is the importance of the annotation @Primary**

When there are multiple beans of the same data-type, developers use the Spring-specific [@Primary annotation](https://examples.javacodegeeks.com/enterprise-java/spring/spring-primary-annotation-example/) that automatically gives the higher preference to a particular bean. This annotation can be used on any class directly or indirectly annotated with the @Component annotation or on methods annotated with the @Bean annotation.

47. What is the difference between the Configuration types XML and Annotation?

Advantages of the annotation:

* All the information is in a single file
* When the class changes, no need to modify the xml file

Advantages of XML file:

* Clear separation between the POJO and its behavior
* When you do not know which POJO is responsible for the behavior, it is easier to find that POJO

48. What is the role of @SpringBootApplication?

The [@SpringBootApplication annotation](https://examples.javacodegeeks.com/enterprise-java/spring/spring-springbootapplication-annotation-example/) was introduced in Spring Boot 1.2.0 and it enables the **auto-configuration** feature.

This annotation encapsulates the working of three different annotations:

* **@Configuration**: Allows the developers to explicitly register the beans
* **@ComponentScan**: Enables the component-scanning so that the controller class and other components will be automatically discovered and registered as beans in spring’s application context
* **@EnableAutoConfiguration**: Enables the auto-configuration feature of spring boot

This annotation takes up the following optional parameters:

* exclude: Excludes the list of classes from the auto-configuration
* excludeNames: Excludes the list of fully qualified class names from the auto configuration
* scanBasePackage: Provides the list of packages which must be applied for scanning
* scanBasePackageClasses: Provides the list of classes in the other package which must be applied for scanning

49. Explain the @InitBinder?

This annotation is decorated on a method in which a date format is declared, and throughout the class, the defined date format is used. Whenever the binding happens with a date field @InitBinder; annotation says to use the CustomDateEditor, which in return uses the date format mentioned in @InitBinder.

50. Define @ControllerAdvice?

Classes with [@ControllerAdvice](https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-controlleradvice-annotation-example/) can be declared explicitly as Spring beans or auto-detected via classpath scanning. All such beans are sorted via AnnotationAwareOrderComparator, i.e. based on @Order and Ordered, and applied in that order at runtime. For handling exceptions, an @ExceptionHandler will be picked on the first advice with a matching exception handler method. For model attributes and InitBinder initialization, @ModelAttribute and @InitBinder methods will also follow @ControllerAdvice order.

51. Can we send an Object as the response of Controller handler method?

Yes we can send JSON or XML based response in restful web services, using the [@ResponseBody annotation](https://examples.javacodegeeks.com/enterprise-java/spring/spring-responsebody-annotation-example/).

52. Explain @ModelAttribute?

[The @ModelAttribute annotation](https://examples.javacodegeeks.com/enterprise-java/spring/spring-modelattribute-annotation-example/) refers to the property of the Model object and is used to prepare the model data. This annotation binds a method variable or the model object to a named model attribute. The annotation accepts an optional value which indicates the name of the model attribute. The @ModelAttribute annotation can be used at the parameter level or the method level. The use of this annotation at the parameter level is to accept the request form values while at the method level is to assign the default values to a model. Let me explain you further with the help of some examples.

53. @RequestMapping annotation

[The @RequestMapping](https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-requestmapping-example/) annotation is used to map the web request onto a handler class (i.e. Controller) or a handler method and it can be used at the Method Level or the Class Level. If developers use the @RequestMapping annotation at a class level, it will be as a relative path for the method level path.

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

[Java based configuration](https://www.javacodegeeks.com/2013/04/spring-java-configuration.html) 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.

55. 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.

56. 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.

E.Spring Data Access

57. Which classes are present in spring JDBC API?

Spring framework provides the following approaches for [Jdbc database access](https://examples.javacodegeeks.com/enterprise-java/spring/spring-jdbctemplate-crud-operations-tutorial/):

* [JdbcTemplate](https://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/)
* SimpleJdbcTemplate
* [NamedParameterJdbcTemplate](https://examples.javacodegeeks.com/enterprise-java/spring/jdbc/jdbc-named-parameters-example-spring-namedparameterjdbctemplate/)
* SimpleJdbcInsert
* SimpleJdbcCall

58. 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 (example [here](http://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/)).

59. JdbcTemplate

JdbcTemplate class provides many convenience methods for doing things such as converting database data into primitives or objects, executing prepared and callable statements, and providing custom database error handling.

60. How can you fetch records by spring JdbcTemplate?

There are two interfaces that can be used to fetch records from the database:

* [ResultSetExtractor](https://examples.javacodegeeks.com/core-java/sql/jdbc-resultsetextractor-example/)
* [RowMapper](https://examples.javacodegeeks.com/enterprise-java/spring/spring-rowmapper-example/)

61. What is the advantage of NamedParameterJdbcTemplate?

[NamedParameterJdbcTemplate](https://examples.javacodegeeks.com/enterprise-java/spring/jdbc/jdbc-named-parameters-example-spring-namedparameterjdbctemplate/) is built upon JDBCTemplate which is provided by spring and used for lower level communication with databases. It makes possible to pass SQL query arguments as key value pairs. As a result the program code is much more readable and therefore serves as better documentation compared to the indexed or the “?” placeholder approach. The latter is harder to follow specially if the number of parameters is huge.

62. What is Spring JDBCTemplate class and how to use it?

[The JdbcTemplate class](https://examples.javacodegeeks.com/enterprise-java/spring/jdbc/spring-jdbctemplate-example/) executes SQL queries, update statements and stored procedure calls, performs iteration over ResultSets and extraction of returned parameter values. It handles the creation and release of resources, thus avoiding errors such as forgetting to close the connection. It also catches JDBC exceptions and translates them to the generic, more informative, exception hierarchy defined in the org.springframework.dao package.

63. What is the difference between JDBC and Spring JDBC?

Spring JDBC value-add provided by the Spring Framework’s on top JDBC layer

1. Define connection parameters
2. Open the connection
3. Specify the statement
4. Prepare and execute the statement
5. Set up the loop to iterate through the results (if any)
6. Do the work for each iteration
7. Process any exception
8. Handle transactions
9. Close the connection

64. 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.

65. What are the ways to access Hibernate by using Spring?

There are two ways to access Hibernate with Spring:

* Inversion of Control with a Hibernate Template and Callback.
* Extending HibernateDAOSupport and Applying an AOP Interceptor node.

66. ORM’s Spring support

Spring supports the following ORM’s:

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

67. How can we integrate Spring and Hibernate using HibernateDaoSupport?

Use Spring’s SessionFactory called LocalSessionFactory. The integration process is of 3 steps:

* Configure the Hibernate SessionFactory
* Extend a DAO Implementation from HibernateDaoSupport
* Wire in Transaction Support with AOP

68. Types of the transaction management Spring support

Spring supports two types of transaction management:

* **Programmatic transaction management:** This means that you have managed the transaction with the help of programming. That gives you extreme flexibility, but it is difficult to maintain.
* **Declarative transaction management:** This means you separate [transaction management from the business code](https://www.javacodegeeks.com/2011/09/spring-declarative-transactions-example.html). You only use annotations or XML based configuration to manage the transactions.

69. 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.

70. 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.

F.Spring Aspect Oriented Programming (AOP)

71. Explain AOP

Aspect-oriented programming, or AOP, is a programming technique that allows programmers to modularize crosscutting concerns, or behavior that cuts across the typical divisions of responsibility, such as logging and transaction management.

72. What are the advantages of spring AOP?

a. It is non-invasive

* Your service/domain classes get advised by the aspects (cross cutting concerns) without adding any Spring AOP related classes or interfaces into the service/domain classes.
* Allows the developer to concentrate on the business code, instead the cross cutting concerns.

b. Its implemented in pure Java

* No need for a special compilation unit, or a special class loader

c. It uses Spring’s IOC for dependency injection

* Aspects can be configured as normal spring beans.

d. As any other AOP framework, it weaves cross cutting concerns into the classes, without making a call to the cross cutting concerns from those classes.

e. Centralize or modularize the cross cutting concerns

* Easy to maintain and make changes to the aspects
* Changes need to be made in one place.
* In one of your classes you don’t want to have logging, it can easily be achieved by modifying the point cut in the respective aspect (logging aspect). So you need to make changes in only one place.

f. Provision to create aspects using schema based (XML configuration) or @AspectJ annotation based style.

g. Easy to configure

73. What are the AOP implementation?

AOP implementations:

* [Spring AOP](https://examples.javacodegeeks.com/enterprise-java/spring/aop/spring-aop-example/):
  + Runtime weaving through proxy is done
  + It supports only method level PointCut
  + It is DTD based
* [Apache AspectJ](https://examples.javacodegeeks.com/enterprise-java/spring/aop/spring-aop-aspectj-example/)
  + Compile time weaving through AspectJ Java tools is done
  + It suports field level Pointcuts
  + It is schema based and Annotation configuration
* JBoss AOP
  + JBoss AOP is not only a framework, but also a prepackaged set of aspects that are applied via annotations, pointcut expressions, or dynamically at runtime. Some of these include caching, asynchronous communication, transactions, security, remoting, and many many more.

74. What are the AOP terminology?

1. Aspect
2. Advice
3. Pointcut
4. JoinPoint
5. Introduction
6. Target Object
7. AOP Proxy
8. Weaving

75. Aspect

The core construct of AOP is the aspect, which encapsulates behaviors affecting multiple classes into reusable modules. It ia a module which has a set of APIs providing cross-cutting requirements. For example, a logging module would be called AOP aspect for logging. An application can have any number of aspects depending on the requirement. In Spring AOP, aspects are implemented using regular classes annotated with the @Aspect annotation (@AspectJ style).

76. Join point

The join point represents a point in an application where we can plug-in an AOP aspect. It is the actual place in the application where an action will be taken using Spring AOP framework.

77. Advice

The advice is the actual action that will be taken either before or after the method execution. This is actual piece of code that is invoked during the program execution by the Spring AOP framework.

Spring aspects can work with five kinds of advice:

* **before:** Run advice before the a method execution.
* **after:** Run advice after the a method execution regardless of its outcome.
* **after-returning:** Run advice after the a method execution only if method completes successfully.
* **after-throwing:** Run advice after the a method execution only if method exits by throwing an exception.
* **around:** Run advice before and after the advised method is invoked.

78. Pointcut

The pointcut is a set of one or more joinpoints where an advice should be executed. You can specify pointcuts using expressions or patterns.

79. What is Introduction?

An Introduction allows us to add new methods or attributes to existing classes.

80. What is Target object?

The target object is an object being advised by one or more aspects. It will always be a proxy object. It is also referred to as the advised object.

81. What is a Proxy?

A proxy is an object that is created after applying advice to a target object. When you think of client objects the target object and the proxy object are the same.

82. What are the different types of AutoProxying?

* BeanNameAutoProxyCreator
* DefaultAdvisorAutoProxyCreator
* Metadata autoproxying

83. What is Weaving? What are the different points where weaving can be applied?

Weaving is the process of linking aspects with other application types or objects to create an advised object.  
Weaving can be done at compile time, at load time, or at runtime.

84. What is the difference between concern and cross-cutting concern in Spring AOP

The Concern is behavior we want to have in a module of an application. A Concern may be defined as a functionality we want to implement.  
The cross-cutting concern is a concern which is applicable throughout the application and it affects the entire application. For example, logging, security and data transfer are the concerns which are needed in almost every module of an application, hence they are cross-cutting concerns.

85. Explain XML Schema-based aspect implementation?

In this implementation case, aspects are implemented using regular classes along with XML based configuration.

86. Explain annotation-based (@AspectJ based) aspect implementation

This implementation case (@AspectJ based implementation) refers to a style of declaring aspects as regular Java classes annotated with Java 5 annotations.

G.Spring Model View Controller (MVC)

87. What is Spring MVC framework?

Spring comes with a [full-featured MVC framework for building web applications](http://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-hello-world-example/). Although Spring can easily be integrated with other MVC frameworks, such as Struts, Spring’s MVC framework uses IoC to provide a clean separation of controller logic from business objects. It also allows to declaratively bind request parameters to business objects.

88. What are the minimum configurations needed to create Spring MVC application?

For creating a simple Spring MVC application, we would need to do the following tasks:

* Add spring-context and spring-webmvc dependencies in the project.  
  Configure DispatcherServlet in the web.xml file to handle requests through spring container.
* Spring bean configuration file to define beans, if using annotations then it has to be configured here. Also we need to configure view resolver for view pages.
* Controller class with request mappings defined to handle the client requests.

89. List out all the concepts that are available in the MVC Architecture?

* The browser sends a request to DispatcherServlet
* DispatcherServlet knows the HanderMapping and can find the appropriate controllers
* Controllers execute the request and put the data in the model and return back the view name to the DispatcherServlet.
* DispatcherServlet uses the view name and ViewResolver to map to the view.

90. DispatcherServlet

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

91. 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.

92. 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.

93. How would you relate Spring MVC Framework to MVC architecture?

Spring MVC framework:

The Spring Framework is an open source application framework and inversion of control container for the Java platform.

MVC architecture:

Model View Controller (MVC) as it is popularly called, is a software design pattern for developing web applications

94. What is ViewResolver in Spring MVC?

Spring provides [ViewResolver](https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-view-resolver-example/), which enable you to render models in a browser without tying you to a specific view technology. Out of the box, Spring enables you to use JSPs, Velocity templates and XSLT views, for example. The two interfaces which are important to the way Spring handles views are ViewResolver and View. The ViewResolver provides a mapping between view names and actual views. The View interface addresses the preparation of the request and hands the request over to one of the view technologies.

95. What is a MultipartResolver and when its used?

Spring MVC provide multipart support with [MultipartResolver](https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-5-servlet-3-0-multipartconfigelement-example/). The MultipartResolver parses inbound multipart requests. You can enable multipart support by registering a MultipartResolver bean in the DispatcherServlet application context.

96. How to upload file in Spring MVC Application?

Spring provides built-in support for [uploading files through MultipartResolver](https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-file-upload-example/) interface implementations. There is also a validator for the field, which will be used to check if the file uploaded is of size greater than zero. There is finally a simple view that contains a form with the option to upload a file.

97. How to validate form data in Spring Web MVC Framework?

There are 3 different ways to [perform validation](https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-form-validation-example/) : using annotation, manually, or a mix of both.

98. What is Spring MVC Interceptor and how to use it?

Spring’s handler mapping mechanism includes handler [interceptors](https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-interceptor-tutorial/), which are useful when you want to apply specific functionality to certain requests, for example, checking for a principal. Interceptors must implement HandlerInterceptor from the org.springframework.web.servlet package. This interface defines three methods:

* preHandle is called **before** the actual handler is executed.
* postHandle is called **after** the handler is executed.
* afterCompletion is called **after the complete request has finished**.

H.Authentication and authorization

99. What is Spring Security?

[Spring security](https://examples.javacodegeeks.com/enterprise-java/spring/spring-security-4-tutorial/) is one of the most important modules of the Spring framework. It enables the developers to integrate the security features easily and in a managed way. In the following example, we will show how to implement Spring Security in a Spring MVC application.

100. Why Spring Boot?

Here are some useful [benefits](https://examples.javacodegeeks.com/enterprise-java/spring/boot/spring-boot-hello-world-example/) of using [Spring Boot](https://www.javacodegeeks.com/spring-boot-tutorials):

* Automatic configuration of an application uses intelligent defaults based on the classpath and the application context, but they can be overridden to suit the developer’s requirements as needed.
* When creating a Spring Boot Starter project, you select the features that your application needs and Spring Boot will manage the dependencies for you.
* A Spring Boot application can be packaged as a JAR file. The application can be run as a standalone Java application from the command line using the java -jar command.
* When developing a web application, Spring Boot configures an embedded Tomcat server so that it can be run as a standalone application. (Tomcat is the default, but you can configure Jetty or Undertow instead.) You can package the application as a WAR file and deploy it to an external servlet container if you prefer
* Spring Boot includes many useful non-functional features (such as security and health checks) right out of the box.

**1) What is a spring?**

Spring is set to be a framework which helps Java programmer for development of code and it provides IOC container, Dependency Injector, MVC flow and many other APIs for the java programmer.

**2) What are Advices in Spring?**

It is the execution of an aspect. Advice is like making your application learn a new trick. They are usually introduced at joinpoints.

**3) What is the default scope of bean in Spring framework?**

The default scope of bean is Singleton for Spring framework.

**4) Name the types of transaction management that are supported by Spring?**

Transaction management supported by Spring are :

* Declarative transaction management.
* Programmatic transaction management.

**5) Is Singleton beans are thread safe in Spring Framework?**

No, singleton beans are not thread-safe in Spring framework.

**6) What are the benefits of Spring Framework?**

Following are the benefits of Spring framework:

* Extensive usage of Components
* Reusability
* Decoupling
* Reduces coding effort by using pattern implementations such as singleton, factory, service locator etc.
* Removal of leaking connections
* Declarative transaction management
* Easy to integrate with third party tools and technologies.

**7) What is Bean Factory?**

Bean Factory is core of the spring framework and, it is a Lightweight container which loads bean definitions and manages your beans. Beans are configured using XML file and manage singleton defined bean. It is also responsible for life cycle methods and injects dependencies. It also removes adhoc singletons and factories.

**8) Define Bean Wiring?**

Bean wiring is the creation of associations between application components that are between the beans in a particular spring container.

**9) What is called Spring MVC?**

A Spring MVC is a single shared controller instance and it is used to handle request type controllers, interceptors which run in the IoC container. It also allows multiple Dispatcher Servlets which can share application context interface but not class based interface.

**10) Why Spring framework is needed?**

Spring framework is needed because it is -

* Very Light Weight Container
* Framework
* IOC
* AOP

**11) Name the various modules used in spring framework?**

* AOP module (Aspect Oriented Programming)
* JDBC abstraction and DAO module
* The Core container module
* MVC framework module
* Application context module
* O/R mapping integration module (Object/Relational)
* Web module

**12) Explain the RowCallbackHandler in Spring?**

The RowCallbackHandler is called for each row in ResultSet and is used to read values from the ResultSet.

**13) Define Application context module?**

This is a very important module and supplies various necessary services like EJB integration, remoting, JNDI access and scheduling. It transforms spring into a framework. It also broadens the idea of BeanFactory by application of lifecycle events, providing support for internationalization messages and validation.

**14) Write about AOP module?**

AOP module is utilized for creating aspects for Spring applications. It also enables support for metadata programming in Spring.

**15) What is a BeanFactory Interface?**

Bean factory interface is used to provide configuration framework for object creation and basic functionality around object management.

**16) State the differences between ApplicationContext and BeanFactory in spring?**

* ApplicationContext allows more than one config files to exist while BeanFactory only permits one.
* ApplicationContext can print events to beans registered as listeners. This feature is not supported by BeanFactory.
* ApplicationContext also provides support for application of lifecycle events, internationalization messages and validation and also provides services like EJB integration, remoting, JNDI access and scheduling. These features too are not supported by Bean Factory.

**17) What is Auto Wiring?**

Autowiring is used to build relationships between the collaborating beans. Spring container can automatically resolve collaborators for beans.

**18) What are the different Modes of Autowiring?**

Autowiring has five different modes:

* no: no autowire
* byName : Autowiring that can be done by property name
* byType : property type as autowired
* constructor: It is similar to byType and it is property is in constructor
* autodetect : Spring is allowed to select autowiring from byType or constructor

**19) How to start using spring?**

Following steps needs to be done to start with the Spring:

* Download Spring and its dependent file from spring's site.
* Create application context xml to define beans and its dependencies
* Integrate application context xml with web.xml
* Deploy and Run the application

**20) What are the methods of bean life cycle?**

There are two important methods of Bean life cycle:

* Setup – called when bean is loaded into container
* Teardown – called when bean is unloaded into container

**21) What are the different types of events of Listeners?**

Following are the different types of events of listeners:

* ContextClosedEvent – This event is called when the context is closed.
* ContextRefreshedEvent – This event is called when context is initialized or refreshed
* RequestHandledEvent – This event is called when the web context handles request

**22) Differentiate between singleton and prototype bean?**

Singleton means only one bean is defined per object instance while Prototype means one definition to more than one object instances in Spring.

**23) What are the types of Dependency Injection?**

Two types of dependency injection are supported by spring framework:

* Setter Injection
* Constructor Injection

**24) Write about Core container module?**

Core container module is responsible for the basic functionality of the spring framework. The whole Spring framework is built with this module as a base.

**25) What is AOP module?**

This AOP module is used for spring enabled application. Support has been provided AOP alliance to ensure the interoperability between spring and other AOP frameworks.

It instructs spring to add annotations to the source code and tell how to apply aspects.

**26) What is AOP Alliance?**

AOP alliance is an open-source project which is aimed at promoting adoption of AOP. The AOP alliance's goal is to define a common set of components and interfaces so as to improve interoperability among different AOP implementations.

**27) What is called spring configuration file?**

Spring configuration file is an XML file and it contains class information. It also describes how these classes are configured and interact with each other.

**28) What are different types of Autowire?**

There are four different types of Auto wire:

* byName
* byType
* constructor
* autodetect

**29) What are the types of the transaction management that is supported by spring?**

Following are the types of transaction management that has been supported by spring:

* declarative
* programmatically

**30) When are declarative and programmatic transaction management used?**

When only a small amount of transactional operations is there, it is advised to use Programmatic transaction management. But if there is a big amount of transactional operations to be taken care of, declarative transaction management is preferred.

**31) What is IOC?**

IOC (Inversion of Control pattern) is also known as dependency injection. IOC directs the programmers to depict how to create objects instead of actually creating them. But in this design pattern, this control has been given to assembler and assembler will instantiate required class if needed.

**32) Write about the different types of Listener related events?**

The different types of events related to listeners are:

* ContextRefreshedEvent – This gets called when the context is refreshed or initialized.
* RequestHandledEvent – This gets called when the web context is handling a request.
* ContextClosedEvent – This gets called when the context gets closed.

**33) What is an Aspect?**

Aspect is also called as logging which is required throughout the application. Logging or aspect is a cross cutting functionality in an application using AOP.

**34) What is a Joinpoint?**

The point where an aspect can be introduced in the application is known as a joinpoint. This point could be a field being modified, a method being called or even an exception being thrown. At these points, the new aspect's code can be added to introduce a new behavior to the application.

Aspect code can be inserted at this point into normal flow of application to change the current behavior.

**35) What is called an Advice?**

Advice will tell application on new behavior and it is the implementation of an aspect. It is inserted into an application at the joinpoint.

Advice is the implementation of an aspect. It is something like telling your application of a new behavior. Generally, the advice is inserted into an application at joinpoints.

**36) What is a Pointcut?**

Pointcut is used to allow where the advice can be applied.

**37) What is weaving?**

Weaving is used to create new proxy object by applying aspects to target object.

**38) What is difference between singleton and prototype bean?**

Singleton Bean - Single bean definition to a single object instance per Spring IOC container

Prototype Bean - Single bean definition to any number of object instances per Spring IOC Container

**39) In what points, can weaving be applied?**

Following are the points where weaving can be applied:

* Compile Time
* Class load Time
* Runtime

**40) What are the different types of AutoProxying?**

Following are the different types of AutoProxying:

* BeanNameAutoProxyCreator
* DefaultAdvisorAutoProxyCreator
* Metadata autoproxying

**41) How can beans be made singleton or prototype?**

The bean tag has an attribute called 'singleton'. The bean is singleton if its value is 'TRUE', otherwise the bean is a prototype.

**42) What classes are used to Control the database connection?**

Following are the classes that are used to control database connection:

* Data Source Utils
* SmartData Source
* AbstractData Source
* SingleConnection DataSource
* DriverManager DataSource
* TransactionAware DataSourceProxy
* DataSource TransactionManager

**43) Describe about DAO in Spring framework?**

DAO is used to provide integration of Java database connectivity and Object relational mapping objects. DAO is spring framework provides connection for JDBC, hibernate, JDO, JPA, Common client interface and Oracle.

**44) What is Autoproxying?**

Autoproxying is used to create proxy automatically for the spring users. It provides following two classes to support this automatic proxy creation:

* BeanNameAutoProxyCreator
* DefaultAdvisorAutoProxyCreator

**45) What is Metadata Autoproxying?**

Metadata Autoproxying can be performed inspiring which can be driven by metadata. This is determined by source level attributes and keeps metadata inside the source code.

This maintains metadata in one place and mainly used for declarative transaction support.

**46) What is 'Throws advice' in Spring?**

'Throws Advice' define the behavior when an exception occurs. It is an interface and it has no methods which need to be implemented.

A class that implements this interface should have method with this signature:

* Void samplethrow (Throw table t)
* Void samplethrow(Method m, Object[] o, Object target, Throw tablet)

**47) What are the various editors used in spring work?**

The various custom editors provided by the Spring Framework are:

* PropertyEditor
* URLEditor
* ClassEditor
* CustomDateEditor
* FileEditor
* LocaleEditor
* StringArrayPropertyEditor
* StringTrimmerEditor

**48) What are the advantages of spring framework?**

Following are the advantages of spring framework:

* Layered Architecture
* Enables Plain Old Java Object (POJO) Programming and it enables continuous integration and testability
* Dependency Injection and Inversion of Control that simplifies JDBC
* Open source framework which can be used for commercial purpose

**49) How is Hibernate accessed using the Spring framework?**

Hibernate can be accessed in the following two ways:

* By IOC with a Callback and HibernateTemplate.
* By applying an AOP Interceptor and broadening the HibernateDaoSupport.

**50) What are the various Channels supported by Spring 2.0?**

Following are the channels supported by spring version 2.0:

* Pollable Channel
* Subscribable Channel
* PublishSubscribe Channel
* Queue Channel
* Priority Channel
* Rendezvous Channel
* Direct Channel
* Executor Channel
* Scoped Channel

**51) Why is declarative transaction management preferred in Spring?**

Declarative transaction management has minimum impact on the application code and, therefore, is an idealistic lightweight container.

**52) Explain the concept of a BeanFactory?**

BeanFactory applies the idea of a factory pattern that utilizes IOC to separate the application's dependencies and configuration from the actual code.

**53) What are the different scopes of spring bean?**

Scopes of spring bean are Singleton, prototype, request, session and global session.

**54) What are all the ways to access Hibernate by using Spring?**

There are two ways to access hibernate using spring:

* Inversion of Control with a Hibernate Template and Callback
* Extending HibernateDAOSupport and Applying an AOP Interceptor node.

**55) How struts application can be integrated with spring?**

There are two options for struts application that can be integrated with spring:

Configuration of Spring to manage beans using ContextLoader plugin and set their dependencies in a spring context file

Grab spring managed beans explicitly using agetwebapplicationcontext()

**56) What is Inversion of control (IOC)?**

Inversion of Control (IOC) is also called as dependency Injection which is nothingbut a design pattern that gives control to the assembler of classes. In general, class will instantiate another class if required.

But in this design pattern, this control has been to given to assembler and assembler will instantiate required class if needed.

**57) Write the benefits of using IOC?**

The major benefits of dependency injection or IOC are that it reduces the amount of coding required for the application. This allows the testing of the application to be done quickly and easily as no JNDI lookup mechanism or singletons are required. IOC containers also support lazy loading and eager installation of services.

**58) What is Inner bean? What is the drawback of inner bean?**

If a bean element is directly embedded in a property tag while wiring beans, then the bean is called Inner Bean. Its drawback is that it cannot be reprocessed.

**59)What are the different types of Injection in spring?**

There are three types of Injection in spring:

* Setter Injection
* Constructor Injection
* Getter or Method Injection

**60) What are the benefits of spring framework?**

Following are the benefits of spring framework:

* Light weight container when compared to j2EE containers
* Built in Web MVC framework
* Creates loosely coupled applications
* Supports aspect oriented programming like logging, transaction and security
* Configuration done in XML format which is easy to write and understand

**61) What are the types of Advice?**

There are five types of Advice:

* Before Advice
* After returning advice
* After throwing advice
* Finally advice
* Around advice

**62) What is called PreparedStatementCreator?**

PreparedStatementCreator is one of the most commonly used interfaces for writing data to the database. createPreparedStatement() is a method that can be used to create and return PreparedStatement from the Connection argument, and exception handling is automatically taken care of. When this interface is implemented, a different interface SqlProvider can also be implemented which has a method called getSql(). This method is useful for providing sql strings to the JdbcTemplate. It does not handle SQLExceptions.

**63) What is SQLProvider?**

SQLProvider has only one method called getSql()and it is implemented using PreparedStatementCreator implementers. It is mainly used for debugging.

**64) Write about BatchPreparedStatementSetter?**

BatchPreparedStatementSetter is used to update more than a single row in one go, they can use BatchPreparedStatementSetter. This interface provides two methods they are

* setValues( PreparedStatement ps, int i) throws SOL exception
* int getBatchSize

**65) What is the better method of using JDBC in Spring?**

If JDBC is used with the template class called JdbcTemplate, it gives a better performance.

**66) What exceptions do the DAO classes, use in Spring throw?**

In spring DAO classes only throws SQLException.

**67) Explain the advantages of using DAO module?**

The database code can be kept clean and simple by using the DAO module. This helps in preventing problems that rise because of poor handling of closures of database resources. Also, the DAO module utilizes the AOP module to enable objects in the Spring application to use transaction management services.

**68) Name the significant ApplicationContext implementations used in the spring framework?**

They are:

* ClassPathXmlApplicationContext
* FileSystemXmlApplicationContext
* XmlWebApplicationContext

**69) How is a bean added to a Spring application?**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN">

<beans>

<bean id="foo"/>

<bean id="bar"/>

</beans>

The bean tag has an ID attribute which stores the bean name and a class attributes which specifies the full class name.

**70) What are ORM integration modules?**

Object/relational mapping (ORM) tool is supported by Spring over straight JDBC by implementing the ORM module. Spring can join various important ORM frameworks, including JDO, iBATIS SQL Maps and Hibernate.

**71) Mention and explain the types of Advice in Spring?**

Types of advice are:

* Before advice: Advice that is executed prior to a joinpoint is called the 'before advice'.
* After returning advice: Advice that is executed after the normal completion of a joinpoint is called the 'after returning advice'.
* After throwing advice: Advice that is executed only if a method exits abnormally by throwing an exception, is called the 'after throwing advice'.
* After (finally) advice: Advice that is executed irrespective of how a joinpoint exits is called 'after finally advice'.
* Around advice: Advice that borders a joinpoint, for example, a method invocation, is called an 'around advice'. This can be used to perform special activities before and after the invocation of method.

**72) What is the web module?**

The web module enables the creation of a web application without XML. The web.xml file needs to be configured for using the web module.

**73) What is DataAccessException?**

DataAccessException is a RuntimeException. It is an Unchecked Exception. The user cannot be forced to handle these kinds of exceptions.

**74) What is XMLBeanFactory?**

Spring includes several applications of Bean factory. Out of these, org.springframework.beans.factory.xml.XmlBeanFactory is a very important one. It loads the beans on the basis of the definitions stored in an XML file. For the creation of an XmlBeanFactory, a java.io.InputStream is passed to the constructor. The InputStream provides the XML to the factory. For example, for retrieval of the bean, the getBean() method is called by passing the name of the desired bean.

MyBean helloBean = (MyBean) factory.getBean("helloBean");

**75) Name the Exception class which is connected to the exceptions thrown by the applications?**

It is the DataAccessException given by org.springframework.dao.DataAccessException

**76) Mention the types of lOC (dependency injection)?**

The different types of loC are: -

* Setter Injection: With the help of JavaBeans properties.
* Constructor Injection: Dependencies are given in the form of constructor parameters.
* Interface Injection: With the help of an interface, an Injection is performed.

Out of these three, only construction and setter are being used in Spring.

**77) What are the important beans lifecycle methods?**

All in all, two bean lifecycle methods are there. The first method is the setup method which is called during the loading of the bean into the container. The second is when the bean is unloaded from the container, and this method is called the teardown.

**78) How can the default lifecycle methods of beans be nullified?**

The tag, bean, has two useful attributes which can be used to define special initialization and destruction methods.

For Example, two new methods forSetup and forTeardown can be added to the Foo class in the following way:

<beans>

<bean id="bar" init-method="forSetup" destroy="forTeardown"/>

</beans>

**79) What is a Target?**

A target is the class that is advised. This class can either be a class to which we want to add a special behavior to or a third party class. The target class is free to center on its major concern using the AOP concepts, regardless of any advice that is being applied.

**80) Explain the term Proxy?**

The term proxy refers to an object which is produced the application of an advice to the target object.

**81) What is cross cutting concern and concern in spring AOP?**

Cross cutting concern: It is a concern which is applicable throughout the application and it affects the entire application. E.g Security, logging and data transfer are the concerns which are needed in almost every module of an application.

Concern: Concern is a behavior that we want to have in a module of an application. Issues in which we are interested defines our concern.

**1. What is Tight Coupling?**

When a class (ClassA) is dependent on another class’s object (ClassB), then we say ClassA is "tightly" Coupled with ClassB. Spring helps us to create classes in a way that Tight Coupling can be removed and Loose Coupling can be done.

**2. What is Loose Coupling?**

Loose Coupling removes the dependency of an object (ClassB) on a class (ClassA). Loose Coupling is approached by creating an interface and a setter & getter method, or by using a constructor which takes the interface object.

**3. What are Beans in Spring?**

When a class is annotated or decorated using the @Component, such a class is called a Bean in Spring. Beans are maintained by Application Context.

**s4. Explain Bean creation process?**

The process of Bean creation has the following phases

(i) Starts with a class (c1) which has the annotation @Component.

(ii) Checks if the component annotated class (c1) is dependent.

(iii) If yes, then Spring will create a bean for that class (c2) too.

(iv) A connection or autowiring will occur between the two classes (c1 and c2) using @Autowired annotation and also through the constructor (c2) or the default case setClass Function (interface the Interface).

**5. What is the importance of the annotation @Primary**

This annotation is used on a class that needs to be taken by spring on a primary basis. For instance, if ClassX is @Component annotated and is dependent on both Class1 and Class2 (both @Component annotated) then the compiler would report an error. To show the primary class between Class1 and Class2 we use @Primary.

**6. What is Dependency Injection?**

Dependency Injection is where Spring searches for beans; once the appropriate bean is found, it autowires the bean to the dependent class. Dependency Injection is the process where Spring framework looks for the beans and identifies the dependencies, and creates the instances of beans and autowires them.

**7. Explain Inversion of Control (IOC).**

In Tight Coupling the dependent class takes the responsibility of creating its dependency. Whereas, in Loose Coupling, we use @Autowired annotation over the dependency class (or reference) and Spring takes control of creating the instance and injects the dependency.

**8. What are the roles of an IOC (Inversion of Control) Container?**

IOC Container does the following things-

 (i) Find Beans

(ii) Identify their dependencies and wire the dependencies

(iii) Manage Lifecycle of the Bean (creation, processing, and destruction)

**9. What is Application Context?**

It is an advanced version of IOC Container. It provides all the functionalities of Bean Factory and also provides things like AOP, Internationalization capabilities, web application context (request, session, etc).

**10. Explain the process of creating an ApplicationContext in Spring.**

The ApplicationContext can be defined in two ways (i) using XML, (ii) using @Configuration. Once the configuration is done in any of the ways defined above, the ApplicationContext is created using new ClassPathXmlApplicationContext. The ClassPathXmlApplicationContext looks for the XML files, using this is one of the two ways. The other way is to use AnnotationConfigApplicationContext.

**11. Explain Component Scan.**

Component Scan is one method of asking Spring to detect Spring-managed components, the input for this search is the packages. Two methods are available to define a Component Scan-

(i) Java Configuration; wherein, we use the @Component annotation to which we specify all the packages, for which Spring does the search.

(ii) XML Configuration- we use <context:component-scan base-package=”com.demo.compscanex”/>

**12. How do you perform the same (above question) in Spring Boot?**

In Spring Boot the annotation used to perform the scan is @SpringBootApplication. This annotation on a class would automatically initiate the component scan on the package where they are in.

**13. Differentiate @Component, @Repository and @Service and @Controller?**

Typically a web application is developed in layers like the controller (which is the initial point of client communication), business (where the actual code or logic of the application is written) and DAO (where the database connections and interaction happens). In such an architecture web application, @Component can be used in any of the layers. Whereas, the @Controller is used in the controller/web layer. @Service is used in the business layer and @Repository is used in the DAO layer.

**14. List out the different scopes of Bean.**

(i) Singleton: throughout the spring context only one instance is created.

(ii) Prototype: a new bean is created whenever requested.

(iii) Request: Every HTTP Request creates a bean.

(iv) Session: A bean for every HTTP Session.

**15. List out the types of Dependency Injection.**

The types of Dependency Injection-

(i) Setter Injection and (ii) Constructor Injection.

**16. What is the difference between the Configuration types XML and Annotation?**

These are the two ways of setting up the configuration, and they perform in the say way. Though, when the annotation approach is taken very less amount of code is written and the result would be the same as compared to the XML approach.

**17. List out the ways Autowiring is done.**

(i) byType  
(ii) byName  
(iii) Constructor (same as byType, but through constructor)

**18. What is Dirty Read?**

When a transaction (t1) is meant to read the changes that are performed by another transaction (t2) and provided transaction t2 is not committed yet; then in such a situation, the transaction t1 is called Dirty Read transaction.

**19. List out the new features available in Spring Framework 4.0 and Spring Framework 5.0?**

Spring 4.0 is the first to support Java features. Spring 5.0 has the support for Reactive Programming and Kotlin.

**20. What is a FrontController?**

In FrontController, the Servlet will not get the first request; the first request would go to FrontController and the request is passed on to the right servlet. In other words, DispatcherServlet is the front controller which intercepts all the requests from the client and then dispatches to appropriate controllers.

**21. What is a ViewResolver?**

ViewResolver enables a web application to select its view (such as JSP) dynamically. ViewResolver gets a name which is appended by /WEB-INF/views and a .jsp. All the display on the content is done in an HTML page.

**22. List out all the concepts that are available in the MVC Architecture?**

(i)  The browser sends a request to DispatcherServlet

(ii) DispatcherServlet knows the HanderMapping and can find the appropriate controllers

(iii) Controllers execute the request and put the data in the model and return back the view name to the DispatcherServlet.

(iv) DispatcherServlet uses the view name and ViewResolver to map to the view.

**23. Explain Model Attribute?**

The annotation @ModelAttribute is decorated on a method typically present inside a Controller. This will help the method to be available in all other methods available in the controller.

**24. What is a Session Attribute?**

The annotation @SessionAttributes (“argument”) is decorated on class (Controller). The attribute (argument) that is present in Model is available in the session.

**25. Explain the @InitBinder?**

This annotation is decorated on a method in which a date format is declared, and throughout the class, the defined date format is used. Whenever the binding happens with a date field @InitBinder; annotation says to use the CustomDateEditor, which in return uses the date format mentioned in @InitBinder.

**26. Define @ControllerAdvice?**

This annotation is used when logic needs to be implemented commonly in multiple classes (Controllers). For instance, if an Exception or its subclasses, or when an exception is raised in the classes, it will be handled by a method annotated with @ExceptionHandler. Whenever an exception occurs in any of the controllers, the exception is handled by the method annotated with @ExceptionHandler.

**27. Why Spring Boot?**

Spring-based applications have a lot of configuration (boiler-plate code). In Spring MVC, a lot of configuration is required (like component scan, dispatcher servlet, view resolver, etc). Whereas, in Spring Boot the boiler-plate code is not required.

**28. Spring vs Spring MVC vs Spring Boot?**

Spring: the most important feature of Spring is Dependency Injection or Inversion of Control.

Spring MVC: provides a decoupled approach in developing web applications. Concepts like DispatcherServlet, ModelAndView, ViewResolver makes web application development easy.

Spring Boot: makes the configuration very easy and automatic using a feature called Auto Configuration, in which the DispatcherServlet is done by Spring internally.

**29. What is the role of @SpringBootApplication?**

This annotation is used to launch up the entire application. Internally, @SpringBootApplication does the following,

@SpringBootConfiguration: same as @Configuration in a Spring Application.

@EnableAutoConfiguration: auto-configures the classes available in the classpath.

@ComponentScan: all the classes available under a package will be scanned when this annotation is applied.

**30. What does an Embedded Server mean in Spring Boot?**

To deploy any web application a server like Tomcat is required. In Spring Boot a server (like Tomcat) is available as part of the application in a jar. The concept of Embedded Server makes the deployment of application very easy and independent.

**31. Why do we use application.properties?**

The file application.properties is used to configure things like database details, log generation, security (username/password), serialization, etc.

**32. What is Spring JDBC?**

Spring JDBC uses methods like update (query), execute (query) and query ([SQL](https://www.greycampus.com/blog/programming/six-ways-sql-and-nosql-databases-offer), resultSetExtractor) to interact with the database.

**33. What is the difference between JDBC and Spring JDBC?**

In JDBC, the checked exceptions need to be written; whereas, in Spring JDBC those exceptions are made into Runtime Exceptions. Which means, exception handling is not manually done in Spring JDBC.

**34. What is JPA?**

Java Persistence API (JPA) defines the mapping from Java Object to a Database Table. The procedure to map a Java object to a row in a database table is defined in JPA. JPA provides a lot of useful annotations, using which the relationship between classes and tables are defined.

**35. What is Hibernate?**  
Once the mapping is done, Hibernate (a JPA Implementation) will help us create query under the hood and interact with the database.

**36. Describe the cases in which the Dependency Injection is done through Constructors and Setters?**

When the dependencies are required/mandatory, the Constructor approach is selected. And when the dependencies are optional then the Setters approach is used.

**37. What is the importance of POM.XML file?**

Project Object Model (POM) is an XML formatted file in which all the configuration for a maven project is defined. The most commonly used tags in POM.XML are <groupid>, <artifactId>, <version>, <packaging> and a few more.

**38. What does the @RequestParam annotation do?**

This allows the server side to read from data and automatically bind it to a parameter coming into the method.

**39. What is Spring Security?**

Spring Security provides security services to J2EE applications. Spring Security is implemented using Servlet Filters under the hood. Servlet Filters are used to pre-process or post-process web requests.

**40. What is CSRF?**

Cross-Site Request Forgery (CSRF) is a security attack where a fraudulent website tricks the user into performing an event on the web application that he/she is logged into. For instance, if the user is logged into the online banking account, this attack tricks the user into transferring the money to an unknown person.