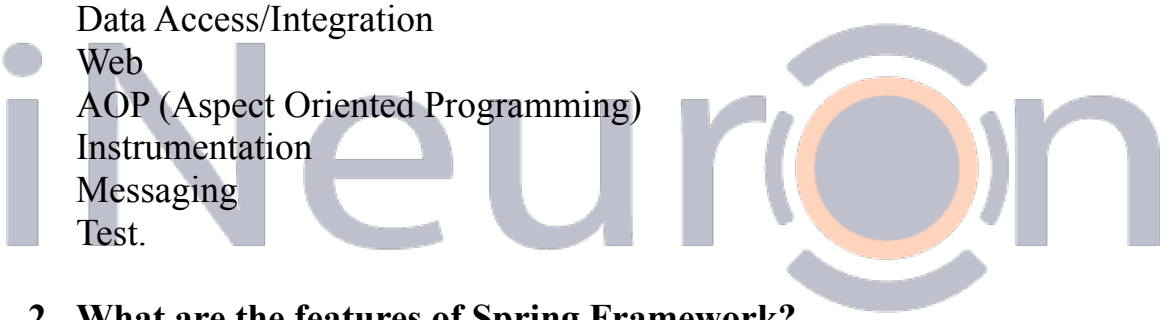
INTERVIEW QUESTIONS - Spring Framework (Part- 1) of 5

1. What is Spring Framework?

Spring is a powerful open-source, loosely coupled, lightweight, java framework meant for reducing the complexity of developing enterprise-level applications.
This framework is also called the “framework of frameworks” as spring provides support to various other important frameworks like JSF, Hibernate, Structs, EJB, etc.
There are around 20 modules which are generalized into the following types:
Core Container
Data Access/Integration
Web
AOP (Aspect Oriented Programming)
Instrumentation
Messaging
Test.



2. What are the features of Spring Framework?

- Spring framework follows layered architecture pattern that helps in the necessary components selection along with providing a robust and cohesive framework for J2EE applications development.
- Spring provides highly configurable MVC web application framework which has the ability to switch to other frameworks easily.
- Provides provision of creation and management of the configurations and defining the lifecycle of application objects.
- Spring has a special design principle which is known as IoC (Inversion ofControl) that supports objects to give their dependencies rather than looking for creating dependent objects.
- Spring is a lightweight, java based, loosely coupled framework.
- Spring provides generic abstraction layer for transaction management that is also very useful for container-less environments.
- Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate or other frameworks) into consistent, unchecked exceptions.

3. **What is a Spring configuration file?**

   A Spring configuration file is basically an XML file that mainly contains the classes information and describes how those classes are configured and linked to each other.

4. **What do you mean by IoC (Inversion of Control) Container?**
   Spring container forms the core of the Spring Framework. The Spring container uses Dependency Injection (DI) for managing the application components by creating objects, wiring them together along with configuring and managing their overall life cycles. The instructions for the spring container to do the tasks can be provided either by XML configuration, Java annotations, or Java code.

5. **What do you understand by Dependency Injection?**

   The main idea in Dependency Injection is that you don't have to create your objects but you just have to describe how they should be created. The components and services need not be connected by us in the code directly.
   We have to describe which services are needed by which components in the configuration file.
   The IoC container present in Spring will wire them up together.

   Java, the 2 major ways of achieving dependency injection are:

   Constructor injection: Here, the IoC container invokes the class constructor with a number of arguments where each argument represents a dependency on the other class.

   Setter injection: Here, the spring container calls the setter methods on the beans after invoking a no-argument static factory method or default constructor to instantiate the bean.

6. **Explain the difference between constructor and setter injection?**

   In constructor injection, partial injection is not allowed whereas it is allowed in setter injection.
   The constructor injection doesn't override the setter property whereas the same is not true for setter injection.
   Constructor injection creates a new instance if any modification is done. The creation of a new instance is not possible in setter injection.
   In case the bean has many properties, then constructor injection is preferred. If it has few properties, then setter injection is preferred.

7. **What are Spring Beans?**
   They are the objects forming the backbone of the user's application and are managed by the Spring IoC container.
   Spring beans are instantiated, configured, wired, and managed by IoC container.
   Beans are created with the configuration metadata that the users supply to the container (by means of XML or java annotations configurations.)

8. **How is the configuration meta data provided to the spring container?**

   XML-Based configuration: The bean configurations and their dependencies are specified in XML configuration files. This starts with a bean tag as shown below:

```
<bean id="student" class="in.ineuron.Student">
 <property name="name" value="NavinReddy"/>
</bean>
```

   Annotation-Based configuration: Instead of the XML approach, the beans can be configured into the component class itself by using annotations on the relevant class, method, or field declaration.
   Annotation wiring is not active in the Spring container by default. This has to be enabled in the Spring XML configuration file as shown below

```
<beans>
<context:annotation-config/>
<!-- bean definitions go here -->
</beans>
```

Java-based configuration: Spring Framework introduced key features as part of new Java configuration support. This makes use of the @Configuration annotated classes and @Bean annotated methods.
Note that:@Bean annotation has the same role as the <bean/> element. Classes annotated with @Configuration allow to define inter-bean dependencies by simply calling other @Bean methods in the same class.

9. **What are the bean scopes available in Spring?**

The Spring Framework has five scope supports. They are:

Singleton: The scope of bean definition while using this would be a single instance per IoC container.
Prototype: Here, the scope for a single bean definition can be any number of object instances.
Request: The scope of the bean definition is an HTTP request.
Session: Here, the scope of the bean definition is HTTP-session.
Global-session: The scope of the bean definition here is a Global HTTP session.
Note: The last three scopes are available only if the users use web-aware ApplicationContext containers.

10. **Explain Bean life cycle in Spring Bean Factory Container.**

The Bean life cycle is as follows:

The IoC container instantiates the bean from the bean's definition in the XML file.
Spring then populates all of the properties using the dependency injection as specified in the bean definition.
The bean factory container calls setBeanName() which take the bean ID and the corresponding bean has to implement BeanNameAware interface.
The factory then calls setBeanFactory() by passing an instance of itself (if BeanFactoryAware interface is implemented in the bean).
If BeanPostProcessors is associated with a bean, then the preProcessBeforeInitialization() methods are invoked.
If an init-method is specified, then it will be called.

Lastly, postProcessAfterInitialization() methods will be called if there are any BeanPostProcessors associated with the bean that needs to be run post creation.

11. What do you understand by Bean Wiring.

When beans are combined together within the Spring container, they are said to be wired or the phenomenon is called bean wiring.
The Spring container should know what beans are needed and how the beans are dependent on each other while wiring beans.
This is given by means of XML / Annotations / Java code-based configuration.

12. What are the different components of a Spring application?

A Spring application, generally consists of following components:

Interface: It defines the functions.
Bean class: It contains properties, its setter and getter methods, functions etc.
Spring Aspect Oriented Programming (AOP): Provides the functionality of cross-cutting concerns.
Bean Configuration File: Contains the information of classes and how to configure them.
User program: It uses the function.

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

Spring Framework can be used in various ways. They are listed as follows:
a. As a Full-fledged Spring web application.
b. As a third-party web framework, using Spring Frameworks middle-tier.
c. For remote usage.
d.As Enterprise Java Bean which can wrap existing POJOs (Plain Old Java Objects).

14. What is autowiring and name the different modes of it?

The IoC container autowires relationships between the application beans. Spring lets collaborators resolve which bean has to be wired automatically by inspecting the contents of the BeanFactory.

Different modes of this process are:

no: This means no autowiring and is the default setting. An explicit bean reference should be used for wiring.

byName: The bean dependency is injected according to the name of the bean. This matches and wires its properties with the beans defined by the same names as per the configuration.

byType: This injects the bean dependency based on type.

constructor: Here, it injects the bean dependency by calling the constructor of the class. It has a large number of parameters.

autodetect: First the container tries to wire using autowire by the constructor, if it isn't possible then it tries to autowire by byType.

15. What are the limitations of autowiring?

Overriding possibility: Dependencies are specified using <constructorarg> and <property> settings that override autowiring.
Data types restriction: Primitive data types, Strings, and Classes can't be autowired

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

BeanFactory: BeanFactory is like a factory class that contains a collection of beans. It instantiates the bean whenever asked for by clients.

ApplicationContext: The ApplicationContext interface is built on top of the BeanFactory interface. It provides some extra functionality on top BeanFactory.

17. Differentiate between BeanFactory and ApplicationContext.

BeanFactory

It is an interface defined in org.springframework.beans.factory.BeanFactory

It uses Lazy initialization

It explicitly provides a resource object using the syntax

It doesn't supports internationalization

It doesn't supports annotation based dependency

ApplicationContext

It is an interface defined in org.springframework.context.ApplicationContext

It uses Eager/ Aggressive initialization

It creates and manages resource objects on its own

It supports internationalization

It supports annotation based dependency

void, if, static, switch, break, continue, new, while, extends, this, super, return……

18. List some of the benefits of IoC.

- It will minimize the amount of code in your application.
- It will make your application easy to test because it doesn't require any singletons or JNDI lookup mechanisms in your unit test cases.
- It promotes loose coupling with minimal effort and least intrusive mechanism.
- It supports eager instantiation and lazy loading of the services.

19. What Is the Default Bean Scope in Spring Framework?

By default, a Spring Bean is initialized as a singleton.

20. Are Singleton Beans Thread-Safe?

No, singleton beans are not thread-safe, as thread safety is about execution, whereas the singleton is a design pattern focusing on creation. Thread safety depends only on the bean implementation itself.

21. What's the difference between @Component, @Controller, @Repository & @Service annotations in Spring??

@Component: This marks a java class as a bean. It is a generic stereotype for any Spring-managed component.The component-scanning mechanism of spring now can pick it up and pull it into the application context.

@Controller: This marks a class as a Spring Web MVC controller. Beans marked with it are automatically imported into the Dependency Injection container.

@Service: This annotation is a specialization of the component annotation. It doesn't provide any additional behavior over the @Component annotation.

You can use @Service over @Component in service-layer classes as it specifies intent in a better way.

@Repository: This annotation is a specialization of the @Component annotation with similar use and functionality.

It provides additional benefits specifically for DAOs.

It imports the DAOs into the DI container and makes the unchecked exceptions eligible for translation into Spring DataAccessException.

22. What do you understand by @Required annotation?

@Required is applied to bean property setter methods. This annotation simply indicates that the affected bean property must be populated at the configuration time with the help of an explicit property value in a bean definition or with autowiring.

If the affected bean property has not been populated, the container will throw BeanInitializationException.

23. What do you understand by @Autowired annotation?

The @Autowired annotation provides more accurate control over where and how autowiring should be done.
This annotation is used to autowire bean on the setter methods, constructor, a property or methods with arbitrary names or multiple arguments. By default, it is a type driven injection.

24. What do you understand by @Qualifier annotation?

When you create more than one bean of the same type and want to wire only one of them with a property you can use the @Qualifier annotation along with @Autowired to remove the ambiguity by specifying which exact bean should be wired.

25. What do you understand by @RequestMapping annotation?

@RequestMapping annotation is used for mapping a particular HTTP request method to a specific class/ method in controller that will be handling the respective request.

This annotation can be applied at both levels:
Class level : Maps the URL of the request
Method level: Maps the URL as well as HTTP request method

26. What are the benefits of using Spring Tool Suite?

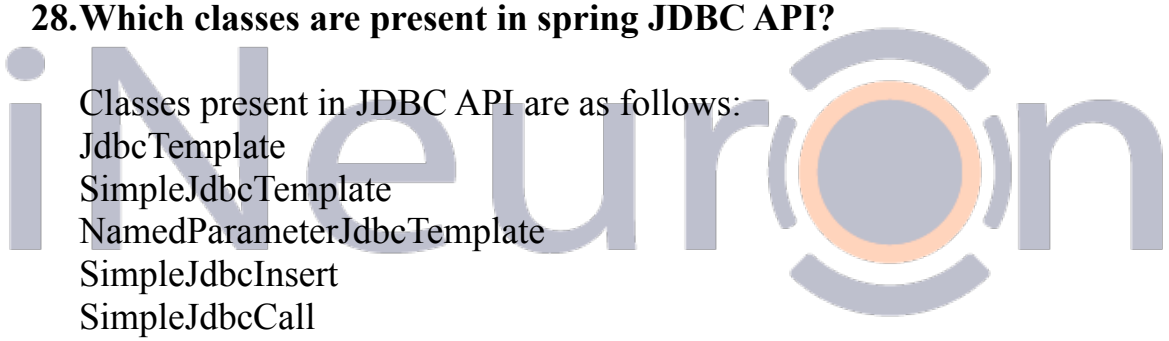
We can install plugins into Eclipse to get all the features of Spring Tool Suite. However, STS comes with Eclipse with some other important kinds of stuff such as Maven support,
Templates for creating different types of Spring projects, and tc server for better performance with Spring applications.
I like STS because it highlights the Spring components and if you are using AOP pointcuts and advice, then it clearly shows which methods will come under the specific pointcut. So rather than installing everything on our own, I prefer using STS when developing Spring-based applications.

27. Name some of the important Spring Modules?

Some of the important Spring Framework modules are:

Spring Context – for dependency injection.
Spring AOP – for aspect-oriented programming.
Spring DAO – for database operations using DAO pattern
Spring JDBC – for JDBC and DataSource support.
Spring ORM – for ORM tools support such as Hibernate
Spring Web Module – for creating web applications.
Spring MVC – Model-View-Controller implementation for creating web applications, web services, etc.

28. Which classes are present in spring JDBC API?



Classes present in JDBC API are as follows:
JdbcTemplate
SimpleJdbcTemplate
NamedParameterJdbcTemplate
SimpleJdbcInsert
SimpleJdbcCall

29. How can you fetch records by Spring JdbcTemplate?

This can be done by using the query method of JdbcTemplate. There are two interfaces that help to do this:
ResultSetExtractor:
It defines only one method extractData that accepts ResultSet instance as a parameter and returns the list.
Syntax:

```
public T extractData(ResultSet rs) throws SQLException,DataAccessException;
```

RowMapper:
This is an enhanced version of ResultSetExtractor that saves a lot of code. It allows to map a row of the relations with the instance of the user-defined class.
It iterates the ResultSet internally and adds it into the result collection thereby saving a lot of code to fetch records

30. What is Hibernate ORM Framework?

Object-relational mapping (ORM) is the phenomenon of mapping application domain model objects to the relational database tables and vice versa.
Hibernate is the most commonly used java based ORM framework.

31. What are the two ways of accessing Hibernate by using Spring?

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

@Controller – for controller classes in Spring MVC project.
@RequestMapping – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through Spring MVC RequestMapping Annotation Examples
@ResponseBody – for sending Object as a response, usually for sending XML or JSON data as a response.
@PathVariable – for mapping dynamic values from the URI to handler method arguments.
@Autowired – for auto wiring dependencies in spring beans.
@Qualifier – with @Autowired annotation to avoid confusion when multiple instances of bean type are present.
@Service – for service classes.
@Scope – for configuring scope of the spring bean.
@Configuration, @ComponentScan, and @Bean – for java based configurations.
AspectJ annotations for configuring aspects and advice, @Aspect, @Before, @After, @Around, @Pointcut, etc.

32. Name some of the design patterns used in Spring Framework?

Spring Framework is using a lot of design patterns, some of the common ones are:
Singleton Pattern: Creating beans with default scope.
Factory Pattern: Bean Factory classes
Prototype Pattern: Bean scopes
Adapter Pattern: Spring Web and Spring MVC
Proxy Pattern: Spring Aspect Oriented Programming support
Template Method Pattern: JdbcTemplate, HibernateTemplate, etc
Front Controller: Spring MVC DispatcherServlet
Data Access Object: Spring DAO support
Dependency Injection and Aspect-Oriented Programming

33. What is HibernateTemplate class?

Prior to Hibernate 3.0.1, Spring provided 2 classes namely: HibernateDaoSupport to get the Session from Hibernate and HibernateTemplate for Spring transaction management purposes. However, from Hibernate 3.0.1 onwards, by using HibernateTemplate class we can use SessionFactory getCurrentSession() method to get the current session and then use it to get the transaction management benefits.
HibernateTemplate has the benefit of exception translation but that can be achieved easily by using @Repository annotation with service classes.

34. Explain the JDBC abstraction and DAO module?

With the JDBC abstraction and DAO module, 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.

35. Explain the object/relational mapping integration module?

Spring also supports using 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.

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

Spring offers the following types of collection configuration elements:

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

37. 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 the Spring framework, which is the JdbcTemplate.

38. What is JdbcTemplate?

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

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

1. It provides a consistent programming model across different transaction APIs such as JTA, JDBC, Hibernate, JPA, and JDO.
2. It provides a simpler API for programmatic transaction management than a number of complex transaction APIs such as JTA.
3. It supports declarative transaction management.
4. It integrates very well with Spring's various data access abstractions.

40. What do you understand by the term 'Spring Boot'?

Spring Boot is an open-source, java-based framework that provides support for Rapid Application Development and gives a platform for developing stand-alone and production-ready spring applications with a need for very few configurations.