Spring Interview Questions

1. What is spring framework?
   Ans) Spring is a light weight, loosely coupled and integration framework for developing enterprise applications easily.

2. What are the differences between EJB and Spring?
   Ans) Enterprise beans vs Java beans

3. How many modules in Spring framework?
   Ans) There are 6 main modules in Spring Framework:
   I. Test module
   II. Core Spring Container
   III. AOP module
   IV. Instrumentation module
   V. Data access & Integration module [Spring-Jdbc, Spring-Hibernate, Spring-txs, etc]
   VI. Web and Remoting [Spring MVC, Spring REST, Spring Security, Spring Struts, etc]

4. What is IOC (or Dependency Injection)?
   Ans) All dependencies are automatically instantiated and injected to bean class i.e., the responsibility is transferred away from bean class and towards dependencies.

5. What are the benefits of IOC (Dependency Injection)?
   Ans) Loose coupling

6. What are the different types of IOCs (dependency injection)?
   Ans) There are 3 types of IOCs among IOC frameworks:
   a) Setter based IOC
   b) Constructor based IOC
   c) Interface dependency

7. What are the types of IOCs supported by Spring?
   Ans) Spring supports 2 types of IOCs: Setter based IOC and Constructor based IOC

8. When to go for Setter based IOC and when to go for Constructor based IOC?
   Ans) Use Constructor based IOC for mandatory dependencies and use Setter based IOC for optional dependencies.

9. What is @Required annotation mean?
   Ans) It is added only on top of setter method to make an **optional setter based IOC** as **mandatory**.

10. What is wiring?
    IOC is wiring i.e., definition of IOC

11. What is the difference between Inversion Of Control (IOC) and Dependency Injection (DI)?
    Ans)
    IOC: It is a design pattern which says Underlying environment is responsible to inject dependencies.
    DI : Implementation of IOC design pattern in spring which says spring container is responsible to inject dependencies.

12. What are features (or advantages)of Spring?
    Ans)
    a) Loose coupling because of DI
    b) Separation because of AOP
    c) Light weight container
    d) Easy to test because of Test module
    e) Predefined Templates
    f) …

13. What is BeanFactory?
    Ans) The BeanFactory means spring container which **lazily** instantiates bean objects after getBean() is invoked at runtime.

14. What is ApplicationContext?
    Ans) The ApplicationContext means spring framework which **eagerly** instantiates bean objects during deployment time without or before invoking getBean() method at runtime.

15. What is the difference between BeanFactory and ApplicationContext?
    Ans) Lazy vs eager

16. What are the types of IOC containers in Spring?
    Ans) There are two types of IOC containers in spring framework.
        a. BeanFactory
        b. ApplicationContext

17. What are different implementation classes of ApplicationContext?
    Ans) FileSystemXmlApplicationContext, ClassPathXmlApplicationContext, AnnotationConfigApplicationContext, XmlWebApplicationContext, etc

18. Can bean id be duplicated?
    Ans) No. Bean id must be unique otherwise throws BeanDefinitionParsingException.

org.springframework.beans.factory.parsing.BeanDefinitionParsingException: Configuration problem: Bean name 'gs1' is already used in this <beans> element.

19. What are bean scopes in spring framework?
    Ans) singleton | prototype | request | session

20. What is singleton?
    Ans) In case of singleton the spring container creates only one bean object even though getBean() with same id is invoked multiple times.

21. What is prototyping?
    Ans) In case of prototyping the spring container creates a separate bean object whenever getBean() is invoked.

22. When prototype bean object is injected to singleton bean, then how many prototype objects are created by container?
    Ans) 1

23. When singleton bean object is injected to prototype bean, then how many single ton objects are created by container?
    Ans) 1

24. What is circular dependency?
    Ans) Circular dependency problem occurred if First bean depends on Second bean and also the Second bean depends on First bean.
    Solution:
    Provide default constructor with setter method but not parameterized constructor.
    Example:

```
package edu.aspire;
public class One {
        private Two two;
        /* public One(){            }  */
        public One(Two t){  this.two = t; }
        /*public void setTwo(Two obj){  this.two = obj;  }*/
}
package edu.aspire;
public class Two {
        private One one;
        /* public Two(){ } */
        public Two(One obj){ this.one = obj; }
        /*public void setOne(One obj){ this.one = obj; }*/
}
<?xml version="1.0" encoding="UTF-8"?>
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.2.xsd">
	<bean id="one" class="edu.aspire.One">
		<constructor-arg>
			<ref bean="two" />
		</constructor-arg>
		<!-- <property name="two">
			<ref bean="two" />
		</property> -->
	</bean>

	<bean id="two" class="edu.aspire.Two">
		<constructor-arg>
			<ref bean="one" />
		</constructor-arg>
		<!-- <property name="one">
			<ref bean="one" />
		</property> -->
	</bean>
</beans>

package edu.aspire.test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class DependencyClient {
	public static void main(String[] args) {
		ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
	}
}
```

25. What is bean life cycle and how it is done?
    Ans) The init-method & destroy-method are called as life cycle methods.

26. What are JSR-250 annotations?
    Spring supports JSR-250 annotations which includes @PostConstruct, @PreDestroy and @Resource annotations.
    @**PostConstruct**: This annotation can be used as an alternate of initialization callback.
    @**PreDestroy**: This annotation can be used as an alternate of destruction callback.
    @**Resource**: This annotation can be used either on top of fields or setter methods but preferably used on top of fields. The @Resource annotation optionally takes either 'name' or ‘type’

attribute. If both attributes are missing then the dependencies are resolved in 'byName' or 'byType' order.

Example:

```
import javax.annotation.Resource;
public class AccountDaoImpl implements AccountDao {
        @Resource(name="bds")
        private DataSource dataSource;

        //public void setDataSource(DataSource dataSource) { this.dataSource = dataSource; }
}

#applicationContext.xml
<beans>
        <bean id="ds"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
                …
        </bean>
        <bean id="bds" class="org.apache.commons.dbcp.BasicDataSource">
                …
        </bean>
</beans>
```

27. What are the differences among @Autowired , @Inject and @Resource?

@Autowired	**@Inject**	**@Resource**
Spring annotation	JSR-330 annotation	JSR-250 annotation
@Qualifier annotation is used to resolve bean ambiguities.	@Named annotation is used to resolve bean ambiguities.	The 'name' attribute is used to resolve bean ambiguities.
Always 'byType' wiring	**Always 'byType' wiring.**	**First 'byName' wiring if not 'byType' wiring.**

28. What is Java Based Configuration in spring? Give some annotation example?

    Ans) Java based configuration option enables us to write most of our Spring configuration without XML but with the help of few Java-based annotations.

    For example:

    @Configuration: Indicates that the class can be used by the Spring IoC container as a source of bean definitions.

    @Bean: Annotation tells Spring that a method annotated with @Bean will return an object that should be registered as a bean in the Spring application context.

29. What is AOP?

    Ans) It says separation between business components and System services (i.e., add on's)

30. What is Aspect, Advice, JointPoint, Pointcut?

Ans) Aspect – Anything add on to business comp.
Advice – Implementation of aspect
JointPoint – The point where aspect is merged is called joinpoint.
Spring supports only method level jointpoint
Pointcut(s) – Selective joinpoint(s) is called as pointcut(s)

31. What are the different types of Advices?
Ans) Before, After, After-returning, Around, After-Throwing

32. What is target?
Ans) The target is a pure business component having only business logic without add-ons.

33. What is a Proxy?
Ans) proxy = target + advice(s)

34. What is weaving?
Ans) The weaving is a process of creating proxy object by merging target and advices.

35. What is Proxy based AOP?
Ans) It was a legacy approach hence it uses classes / interfaces from Spring API.

36. What is Declarative based AOP?
Ans) It was newly added to Spring 2.5 version and uses Binding annotations.

37. What is Annotation based AOP?
Ans) It was newly added to Spring 2.5 and uses Java annotations.

38. What are the Advantages with AOP?
Ans) a) Business components are clean
b) Enhances maintainability.
c) Promotes re-usability.
d) Clear demarcation means separation among developers in team.

39. What is autowiring?
Ans) Autowiring enables the container to inject the beans automatically. The advantage with autowiring is reduces spring configuration file size by eliminating <property> and <constructor-arg> elements.

40. What are the modes of autowiring?
Ans) "no | byname | byType | constructor"

41. What is Annotation based wiring?
Ans) 6 rules

42. How to enable (or activate) annotation based wiring in spring container?
    Ans) Add <context:annotation-config/> in spring configuration file.

43. What is Autodiscovery?
    Ans) 4 rules

44. How to enable (or activate) autodiscovery?
Ans) Add <context:component-scan/> in spring configuration file.

45. How to avoid bean element ambiguities?
    Ans) Using '@Qualifier' annotation

46. What are Spring Annotations?
    Ans) @Autowired, @Qualifier, @Required, @Component, @Aspect, @PointCut, @Controller, @RequestMapping, etc

47. What are JSR (sun) annotations?
    Ans) @Inject, @Named, @Resource, @PostConstruct, @PreDestroy, etc

48. What are the different types of DataSources supported by Spring?
    Ans) Spring framework supports 3 types of datasources:
    - **a)** Built-in dataSource
    - **b)** Third party datasource
    - **c)** JNDI datasource

49. How can you configure a bean to get DataSource from JNDI?
    Ans) Using **JndiObjectFactoryBean** class and its property 'jndiName'. The value should be java:/comp/evn/mypool

50. What is DataAccessException?
    Ans) All exceptions from all persistence mechanisms are wrapped into common set of spring specific exception types with **DataAccessException** is root of the hierarchy.

51. What is Spring-Jdbc?
    Ans) The Spring-Jdbc module contains JdbcTemplate class and PreparedStatementCreator interface.

52. What is Spring-Hibernate?
    Ans) The Spring-Hibernate module contains HibernateTemplate class and HibernateCallback interface.

53. What is JdbcTemplate?
    Ans) This class automatically provides fixed steps in jdbc code.

54. What is PreparedStatementCreator?
    Ans) It is a callback interface which contains createPreparedStatement() callback method to write variable steps.

55. What is HibernateTemplate?
    Ans) This class automatically provides fixed steps in hibernate code.

56. What is DaoSupport?
    Ans) The purpose of DaoSupport class is to eliminate template property from source code (DAO impl class) as well as eliminate <bean> element for Template class from spring configuration file.

57. What is HibernateDaoSupport?
    Ans) The HibernateDaoSupport class contains HibernateTemplate and SessionFactory properties.

58. How spring configuration file refers properties files?
    Ans) Using **PropertyPlaceholderConfigurer**

59. What are the different types of transactions supported by Spring?
    Ans) Spring supports both programmatic and declarative transactions.

60. What are the differences between Programmatic and Declarative transactions?
    Ans)

Programmatic Transactions	**Declarative Transactions**
Business component contains both database operations as well as transactional statements.	Business component contains only database operations without transactional statements.
Gives fine grained control since single method may have multiple transactions.	Convenient to use because of AOP.

61. Why most of the spring users choose declarative transaction management?
    Ans) In case of declarative transactions:
    a) The business comp contains only db operations without transactional statements
    b) The Transactional statements are part of transaction aspect
    c) The above transaction aspect is automatically provided by spring container
    d) Our business comp and container provided transaction aspect is merged using **AOP**.

62. What are the Propagation Behaviors (transaction attributes) supported in spring?
    Ans) Spring framework supports 7 transactional attributes: Required, RequiresNew, Supports, NotSupported, Mandatory, Never, Nested

63. What are the Isolation Levels in Spring?

Ans) ISOLATION_READ_UNCOMMITTED, ISOLATION_READ_COMMITTED, ISOLATION_REPEATABLE_READ, ISOLATION_SERIALIZABLE, ISOLATION_DEFAULT

64. Explain Spring MVC Flow?

Ans)

I. DispatcherServlet receives request and delegates request to controller class with the help of HandlerMapping.
II. DispatcherServlet receives ModelAndView object from controller class and gets logical name from it.
III. DispatcherServlet consults with ViewResolver to map between logical name with corresponding physical response page name.
IV. Finally, DispatcherServlet will render (means display) response with the help of view object.

65. What is DispatcherServlet?

Ans) It is the name of front controller.

66. What is controller class?

Ans) Controller class means business component.

67. What is HandlerMapping?

Ans) It is used to map request page with corresponding controller class using urlpath.

68. What is InternalResourceViewResolver?

Ans) It is used to map logical name with corresponding physical response page name. Also, it selects corresponding view class.

69. What is View implementation?

Ans) Spring API contains multiple views to render means display response in appropriate format depending on client preference such as xml, json, html, etc.
Example: InternalResourceView, JstlView, TilesView, VelocityView, FreeMarkerView, MappingJacksonJsonView, etc.

.

70. What are the new features in Spring 2.5, Spring 3.0 and Spring 4.0?

Ans)

SPRING 2.5 Features

a) Supports Spring annotations
b) Supports XSD validation
c) Test module was newly added to write spring test cases easily
d) Declarative based AOP and Annotation based AOP
e) Annotation based wiring and Autodiscovery
f) Annotation based controller

g) …

SPRING 3.0 Features

a) Additionally supports SUN (JSR) annotations
b) Supports RESTFul web services
c) JSR-330 Inject model
d) Java-based configuration

SPRING 4.0 Features

a) Spring Boot
b) Leverages Java 8 Features such as Lambda expressions, Functional Programming, etc
c) RestController
d) …

71. How to protect or restrict .jsp pages in Spring MVC applications?

Ans) Place .jsp files anywhere inside WEB-INF folder.

72. Which design patterns were used by your previous spring project?

Ans)

a) IOC design pattern
   It is a design pattern which says Underlying environment is responsible to inject dependencies.

b) Factory design pattern
   Create object without exposing the creation logic to the client and refer to newly created object using a common interface.
   Example:
   Public **Object** getBean(String **id**) //it is factory method

c) Singleton design pattern
   Create an object while making sure that **only single object gets created**.
   Spring container creates only one bean object even though getBean() method with same id is invoked multiple times.

d) Prototype design pattern
   Prototype pattern refers to creating duplicate objects while keeping performance in mind. This pattern involves implementing a prototype interface which tells to create a ***clone*** of the current object. This pattern is used when creation of object directly is costly. We can cache the object, returns its clone on next request.

e) Proxy pattern
   Merging business component and services.

Proxy = target + advice(s)

f) DAO Design Pattern
For every table, we have to write a separate Model class, Dao interface and Dao Impl class.

g) Template callback design pattern
This design pattern is used to avoid boiler plate code i.e., avoid fixed steps i.e., reduce code. In this design pattern there must be template class and at least one call back interface.

h) MVC design pattern

i) Front controller design pattern
Write only one controller class so that application controller is centralized.
For example, spring provides "DispatcherServlet" to ensure that an incoming request gets mapped with corresponding controller class means business class.

j) Service locator design pattern
The service locator design pattern is used when we want to locate various services using JNDI lookup. Considering high cost of looking up JNDI for a service, Service Locator pattern makes use of **caching** technique. For the first time a service is required, Service Locator looks up in JNDI and caches the service object. Further lookup of same service via Service Locator is done in its cache which improves the performance of application to great extent.

k) View helper
Spring has number of custom JSP tags and velocity macros to assist in separating code from presentation in views.

73. How to load multiple spring configuration files in web layer?

Ans) The **ContextLoaderListener** is a servlet listener that loads additional configuration files into spring application context along with application context created by DispatcherServlet.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <servlet>
        <servlet-name>disp</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
            <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
            <servlet-name>disp</servlet-name>
```

```
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/RootConfig.xml </param-value>
    </context-param>
</web-app>
```

Other Interview question Links

http://www.tutorialspoint.com/spring/spring_interview_questions.htm
http://www.javatpoint.com/spring-interview-questions
https://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html
http://www.mkyong.com/tutorials/spring-tutorials/
http://www.javacodegeeks.com/2014/05/spring-interview-questions-and-answers.html
http://www.journaldev.com/2696/spring-interview-questions-and-answers#spring-overview