Spring Quiz

Question 1: IOC or Dependenct injection is a ...

1. **Design Pattern**
2. Framework
3. Java Module
4. ORM Framework

Question 2: Controller in Spring is ....

1. a abstract class
2. concrete class
3. final class
4. **an interface**

Question 3: Spring is ....

1. a free framework
2. **an open source framework**
3. a licensed framework
4. a properietary framework

Question 4: Which are different points where weaving can be applied?

1. Compile Time and Classload Time
2. Compile Time , Classload Time , load time
3. Compile Time and Runtime
4. **Compile Time , Classload Time , Runtime**

Question 5: Beans defined in spring framework are by default ...

1. Abstract
2. **Singleton**
3. Final
4. initialized

Question 6: Which exception class is related to all the exceptions that are thrown in spring applications?

1. ArrayIndexOutofBound
2. **DataAccessException**
3. NullPointerException
4. SpringException

Question 7: What are Different Autowire types ...?

1. byName , byType, destructor and autodetect

2. byName , byMethod, constructor and autodetect
3. byName , byType, constructor and autocorrect
4. **byName , byType, constructor and autodetect**

Question 8: Which of the following is not a Spring Module?

1. AOP
2. O/R Integration
3. Spring MVC
4. **HTML/JSP**

Question 9: What are different types of Bean Injections?

1. **constructor and setter**
2. constructor and getter
3. getter and setter
4. setter, getter and constructor

Question 10: Which of the following is true?

1. ApplicationContext implements BeanFactory
2. **ApplicationContext extends BeanFactory**
3. BeanFactory extends ApplicationContext
4. BeanFactory implements ApplicationContext

Question 11: Choose which is not true:

1. Central to the Spring Framework is its IOC container
2. IOC provides a consistent means of configuring and managing Java objects using call-backs.
3. IOC container is responsible for managing object lifecycles
4. **All of above**

Question 12: Choose correct option:

1. The first version was written by Rod Johnson who released in October 2002.
2. The framework was first released under the Apache 2.0 license in June 2003.
3. The Spring 1.2.6 framework won a Jolt productivity award and a JAX Innovation Award in 2006.
4. **1,2,3**

Question 13: What is the Type of Proxy in Spring Framework?

1. one
2. **two**
3. three
4. four

Description: 1. static
2. Dynamic

Question 14: What is stand for Spring MVC?

1. **Model view Controller**
2. Middle view Controller
3. Module view Controller
4. none

Question 15: How many type of the injection method in Spring?

1. one
2. **two**
3. three
4. four

Question 16: How many type of Dynamic proxy is available in Spring?

1. one
2. three
3. **two**
4. none

Description: 1. JDK Dynamic Proxy
2. CGLIB Dynamic Proxy

Question 17: Choose correct option:

1. The Spring Framework is an open source application.
2. The Spring is in the Java platform.
3. The Spring is in for .NET Framework.
4. **1,2,3**

Question 18: Spring MVC is a request-based framework.

1. **Yes**
2. No

Question 19: What is the Stand for AOP?

1. **Aspect Oriented Programming**
2. Aspect Oriented Programs
3. Aspect Oriented Programming
4. None

Question 20: Spring provides features:

1 .Resource management
2 .Exception handling
3 .Transaction participation
4 .Resource unwrapping
5. Abstraction for BLOB and CLOB handling

1. 1,2,3,4,5
2. **1,2,3,4**
3. 1,2,4,5
4. None

Question 21: Given the following Spring configuration file, what is the correct answer:
<bean class="com.spring.service.MyServiceImpl">
<property name="repository" ref="jpaDao"/>
</bean>
<bean class="com.spring.repository.JpaDao"/>

1. The first declared bean MyServiceImpl is missing an id must be named myService
2. **The second declared bean JpaDao is missing an id must be named jpaDao**
3. Answers 1 and 2 are both rights
4. Answers 1 and 2 are both wrong

Description: Those beans are anonymous because no id is supplied explicitly. Thus Spring container generates a unique id for that bean. It uses the fully qualified class name and appends a number to them. However, if you want to refer to that bean by name, through the use of the ref element you must provide a name. To be correct, the 2nd bean has to declare a jpaDao id attribute in order to be reference by the repository property of the first bean.

Question 22: What statement is not correct in live environment? Select a unique answer.

1. Constructor and properties autowiring in the same bean are not compatible
2. A bean should have a default or a no-args constructor
3. The <constructor-arg> tag could take type, name and index to reduce ambiguity
4. **All of the above**

Description: 1. You may auto-wiring properties by constructor, setter or properties in the same bean
2. The <constructor-arg> tag helps to instanciated a bean without default or no-args constructor
3. The <constructor-arg> tag could take type and index to reduce ambiguity, but not name which requires debug symbols.

Question 23: What is/are typically case(s) where you usually need to manually instantiated an ApplicationContext?

1. In a web application
2. In an integration test running with the SpringJUnit4ClassRunner
3. **In a standalone application started with a main method**
4. None of the above

Description: 1. In a web application, the ContextLoaderListener is in charge to create an WebApplicationContext.
2. In an integration test based on Spring, the SpringJUnit4ClassRunner creates the application context for you. The @ContextConfiguration annotation allows to specified application context configuration files.
3. In a main method, you have to instanciated a class implementing the ApplicationContext interface (examples: ClassPathXmlApplicationContext or FileSystemXmlApplicationContext)

Question 24: How to auto-inject into a field a bean by its name? Select one or more response.

1. With the name attribute of the @Autowired annotation
2. By using the single @Qualifier annotation
3. **By using both the @Autowired and the @Qualifier spring annotations**

Description: page 179

Question 25: Select the right statement about referring a Spring configuration file inside the package
com.example.myapp in the below example?
ApplicationContext context = new
ClassPathXmlApplicationContext("classpath:/com.example.myapp.config.xml");

1. The classpath: prefix could be omit
2. Package name with dot is not well formatted using the dot character
3. The slash character preceding com.example could be omit
4. **All of the above**

Description: 1. When using the ClassPathXmlApplicationContext, the classpath: prefix is default one so you could omit it
2. In a Spring location resource, package separator is a slash and not a dot. Thus the com/example/myapp/config.xml syntax has to be used.
3. ClassPathXmlApplicationContext starts looking from root of the classpath regardless of whether specify "/"

Question 26: How could you externalize constants from a Spring configuration file or a Spring annotation into a .properties file? Select one or more answers

1. By using the util:constant tag
2. By declaring the ConstantPlaceholderConfigurer bean post processor
3. **By using the context:property-placeholder tag**
4. By using the c: namespace

Description: 1. The <util:constant static-field="constant name"/> tag enables to reference a Java constant or
enumeration into a spring configuration file
2. ConstantPlaceholderConfigurer does not exist. You may think about the PropertyPlaceholderConfigurer bean post processor.
3. The <context:property-placeholder location="file:/myApp.properties" /> tag activates the replacement of ${...} placeholders, resolved against the specified properties file.
4. The c: namespace is for simplifying constructor syntax (since Spring 3.1) and don´t provide such feature.

Question 27: Given the Spring configuration file, which are the correct statements?
<bean class="com.spring.service.BankServiceImpl"
p:bankName="NationalBank">
</bean>

1. **The p namespace has to be declared**
2. Bean id is bankServiceImpl
3. The BankServiceImpl references a NationalBank bean

Question 28: What one is not the right affirmations about the @PostConstruct, @Resource and the @PreDestroy annotations?

1. Those annotations are specified in the JSR-250
2. The context:component-scan tag enable them
3. **The Spring Framework embedded those annotation**
4. The context:annotation-config tag enable them

Question 29: What are the main advantages of using interfaces when designing business services? Select answer.

1. **Mocking or stubbing the service**
2. Be able to use the Spring auto-injection
3. Can do dependency checking

Description: 1. With modern mock API like Mockito or EasyMock, interfaces are not mandatory for mocking or stubbing the service. But using interface remains easier when you have to manually mock the service in unit test.
2. Auto-injection is possible with class. Spring uses CGLIB.
3. Dependency checking is an advantage of dependencies injection.

Question 30: How is named the bean that is defined in the following configuration class. Select a single answer.

```
@Configuration
public class ApplicationConfig {
@Autowired
private DataSource dataSource;
@Bean
ClientRepository clientRepository() {
ClientRepository accountRepository = new JpaClientRepository();
accountRepository.setDataSource(dataSource);
return accountRepository;
}
}
```

1. JpaClientRepository
2. **clientRepository**
3. Two beans are defined : a data souce and a repository

Description: The @Bean annotation defines a String bean with the id clientRepository. JpaClientRepository is the implementation class of the bean. The data source is injected and is not declared in this class.

Question 31: What is/are typically case(s) where you usually need to manually instantiated an ApplicationContext?

1. In a web application
2. In an integration test running with the SpringJUnit4ClassRunner
3. **In a standalone application started with a main method**
4. None of the above

Question 32: Select correct statement about developing integration test with Spring support.

1. A new Spring context is created for each test class
2. To get a reference on the bean you want to test, you have to call the getBean() method of the Spring context
3. **Spring context configuration could be inherits from the super class**
4. The Spring context configuration file has to be provided to the @ContextConfiguration annotation

Description: 1. The Spring context is cached across tests unless you use @DirtiesContext annotation
2. With the Spring test module, dependency injection is available in test case. So you may autowired the bean to test
3. By default, a @ContextConfiguration annoted class inherits the spring context configuration file locations defined by an annotated superclass. The inheritLocations of this attribute allows to change this default behavior.
4. If no context configuration file is provided to the @ContextConfiguration annotation, Spring use a file convention naming. It try to load a file named with the test class name and suffices by the "-context.xml" suffice (i.e. MyDaoTest-context.xml)

Question 33: Select one correct answer about spring bean life cycle.

1. The method annoted with @PostConstruct is called after bean instantiation and before properties setting of the bean
2. The method @PreDestroy of a prototype bean is called when the bean is garbage collected
3. The init() method declared in the init-method attribute of a bean is called before the afterPropertiesSet callback method of the InitializingBean interface
4. **The method annotated with @PostConstruct is called before, before the afterPropertiesSet callback method of the InitializingBean interface**

Description: 1. In the bean lifecycle, method annotated with @PostConstruct is called after the properties set step and the BeanPostProcessors#postProcessBeforeInitialization step
2. Destroy methods of prototype beans are never called
3. In the bean lifecycle, the afterPropertiesSet callback method of the InitializingBean is called after the method annotated with the @PostConstruct annotation and before the init-method declared in the XML configuration file.
4. In the bean lifecycle, the method annotated with the @PreDestroy annotation is called before the destroy callback of the DisposableBean interface and before the destroy-method declared in the XML configuration file.

Question 34: What is right about the spring test module?

1. It provides an abstraction layer for the main open source mock frameworks
2. Provides the @Mock annotation
3. It dynamically generates mock objects
4. **None of the above**

Question 35: Given the following configuration class, what are correct affirmations? Select one or more answers.
public class ApplicationConfig {
private DataSource dataSource;
@Autowired

```
public ApplicationConfig(DataSource dataSource) {
this.dataSource = dataSource;
}
@Bean(name="clientRepository")
ClientRepository jpaClientRepository() {
return new JpaClientRepository();
}
}
```

1. **@Configuration annotation is missing**
2. @Bean name is ambiguous
3. @Bean scope is prototype

Description: 1. In order to be taken into account by Spring, the ApplicationConfig class has to be annotated with the @Configuration annotation

Question 36: What are the main advantage(s) for using Spring when writing unit tests?

1. Reuse Spring configuration files of the application
2. Use dependency injection
3. **Provide some mocks for servlet classes**
4. All of the above

Description: What are the main advantage(s) for using Spring when writing unit tests?
1. You don´t need Spring container to writer unit test
2. Refer to the answer number 1.
3. The org.springframework.mock package provides mock classes like MockHttpSession or MockHttpContext. They could be helpful for unit test in the presentation layer and when you don´t use any mock framework such as Mockity or EasyMock.

Question 37: Select correct statement about transactional support of the spring test module.

1. **Transaction manager could be set within the @TransactionConfiguration annotation**
2. Method annotated with @Before is executed outside of the test´s transaction
3. Spring test may rollback the transaction of a service configured with the REQUIRES_NEW propagation

Question 38: Which one is not correct about the advantages for using Spring when writing integration tests?

1. Reuse Spring configuration files of the application
2. **Create mock or stub**
3. Be able to use the rollback after the test pattern
4. Use dependency injection

Description: Mocking or stubbing is more frequent in unit tests than in integration tests. And Spring does not provide any implementation or abstraction of mock framework.

Question 39: What are the features of the XML <context: namespace? Select correct option

1. @Transactional annotation scanning
2. @Aspect annotation detection enabling
3. **@Autowired annotation enabling**

Description: 1. Use <tx:annotation-driven /> to enable @Transactional annotation scanning
2. Use <aop:aspectj-autoproxy /> to enable detection of @Aspect bean
3. Turns on <context:annotation-config /> or <context:component-scan /> to enable @Autowiring annotation

Question 40: What statement is not correct in live environment? Select a unique answer.

1. Constuctor and properties autowiring in the same bean are not compatible
2. A bean should have a default or a no-args constructor
3. The constructor-arg tag could take type, name and index to reduce ambiguity
4. **All of the above**

Question 41: Which one is the correct statement about AOP proxy?

1. **AOP proxies are created by Spring in order to implement the aspect contracts**
2. AOP proxies are always created with a JDK dynamic proxy
3. Only classes that implements at least one interface could be proxied
4. All methods could be proxied

Description: 1. An object created by the AOP framework in order to implement the aspect contracts
2. If the target object does not implement any interfaces then a CGLIB proxy will be created. You could also use CGLIB proxy instead of JDK dynamic proxy
3. If the target object does not implement any interfaces then a CGLIB proxy will be created.
4. When CGLIB proxy is used, final methods cannot be advised, as they cannot be overridden.

Question 42: Select method´s signatures that match with the following pointcut: execution(* com.test.service..*.*(*))

1. **void com.test.service.MyServiceImpl#transfert(Money amount)**
2. void com.test.service.MyServiceImpl#transfert(Account account, Money amount)
3. void com.test.service.account.MyServiceImpl#transfert(Account account, Money amount)

Question 43: What is an after returning advice? Select a unique answer.

1. Advice to be executed regardless of the means by which a join point exits
2. Advice that surrounds a method invocation and can perform custom behavior before and after the method invocation
3. Advice to be executed before method invocation
4. **Advice to be executed after a join point completes without throwing an exception**

Question 44: What are the unique correct answers about Spring AOP support?

1. An advice could proxied a constructor?s class
2. **A point cut could select methods that have a custom annotation**
3. Static initialization code could be targeted by a point cut
4. Combination of pointcuts by &&, || and the ! operators is not supported

Description: The @annotation designator enables to select methods that are annotated by a given annotation

Question 45: What is an after throwing advice? Select a unique answer.

1. Advice that could throw an exception
2. **Advice to be executed if a method exits by throwing an exception**
3. Advice that executes before a join point
4. Spring does not provide this type of advice

Description: 1. A before advice could throw an exception
2. An after throwing advice is executed if a method exits by throwing an exception
3. An advice that executes before a join point is named a before advice
4. Spring supports after throwing advices

Question 46: What is an advice? Select a unique answer.

1. **An action taken by an aspect at a particular join point**
2. A point during the execution of a program
3. An aspect and a pointcut
4. A predicate that matches join points

Question 47: What is a pointcut?

1. Code to execute at a join point
2. **An expression to identify joinpoints**
3. An advice and a jointpoint
4. None of the above

**Question 48: Using the Spring AOP framework, what are the joinpoint methods of the following pointcut
expressions?
execution(public * *(..))**

1. **The execution of all public method**
2. The execution of all public method returning a value
3. The execution of all public method having at least one parameter
4. The execution of all public method in class belonging to the default java package

**Question 49: Considering 2 classes AccountServiceImpl and ClientServiceImpl. Any of these 2 classes inherits from
each other. What is the result of the pointcut expressions?
execution(* *..AccountServiceImpl.update(..))
&& execution(* *..ClientServiceImpl.update(..))**

1. Matches pubic update methods of the 2 classes, whatever the arguments
2. Matches any update methods of the 2 classes , whatever the arguments and method visibility
3. Matches any update methods of the 2 classes , with one more arguments and whatever method visibility
4. **No joint point is defined**

Description:
Poincut expression could not satisfied both first and second execution point. Do not confuse the && operator and || operator.

**Question 50: Using the Spring AOP framework, what is the visibility of the method matches by the following join
point?
@Pointcut("execution(* *(..))")
private void anyOperation() {};**

1. All methods, whereas there visibility
2. All methods, except private method
3. Protected and public methods
4. **Public methods**

Description: Due to the proxy-based nature of Spring´s AOP framework, protected methods are by definition not intercepted, neither for JDK proxie nor for CGLIB proxies. As a consequence, any given pointcut will be matched against public methods only! To intercept private and protected methods, AspecJ weaving should be used instead of the Spring´s proxy-bases AOP framework.