Question

Container

Question 1

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

Question 2

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
4. NationalBank is a scalar value

Question 3

What the name of the bean 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. jpaClientRepository
3. clientRepository
4. Two beans are defined: a data souce and a repository

Question 4

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

Question 5

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. None of the above
5. All of the above

Question 6

What are the right affirmations about the @PostConstruct, @Resource and the @PreDestroy annotations?

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

Question 7

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 8

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 omitted
2. Package name using the dot character is not well formatted
3. The slash character preceding com.example could be omit
4. All of the above
5. None of the above

Question 9

How to auto-inject into a field a Spring bean by its name? Select one or more answer choices.

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
4. By using the @Autowired annotation and naming the field with the bean name

Question 10

What are the main advantages of using interfaces when designing business services? Select one or more answer choices.

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

Question 11

Select one or many correct answers about Spring bean life cycle.

1. The method annotated 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 the afterPropertiesSet callback method of the InitializingBean interface

Question 12

Given the following configuration class, what are the 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. Default or no-arg constructor is missing
3. @Bean name is ambiguous
4. @Bean scope is prototype

Question 13

What are the features of the XML <context:namespace? Select one or many answers.

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

Test

Question 14

Select one or more correct statements 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 inherited from the super class
4. The Spring context configuration file has to be provided to the @ContextConfiguration annotation

Question 15

What are the main advantage(s) 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

Question 16

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
5. None of the above

Question 17

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. All of the above
5. None of the above

Question 18

Select correct statement(s) 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
4. The transaction of a method annotated with the @Rollback annotation with its default values is rolled back after the method has completed

AOP

Question 19

Considering 2 classes AccountServiceImpl and ClientServiceImpl. Any of these 2 classes inherits from each other. What is the result of the following pointcut expression?

```
   execution(* *..AccountServiceImpl.update(..))
&& execution(* *..ClientServiceImpl.update(..))
```

1. Matches public 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

Question 20

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 their visibility
2. All methods, except private method
3. Protected and public methods
4. Public methods

Question 21

What are the 2 correct statements 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 a least one interface could be proxied
4. All methods could be proxied
5. Proxies are created by a BeanPostProcessor

Question 22

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

Question 23

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 24

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 25

What is a pointcut? Select the single answer.

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 26

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(Money amount)
4. void com.test.service.account.MyServiceImpl#transfert(Account account, Money amount)
5. None of the above

Question 27

What are the unique right answer about Spring AOP support?

1. An advice could proxied a constructor's class
2. A pointcut 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

Question 28

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

Data Access

Question 29

Why is it a best practice to mark transaction as read-only when code does not write anything to the database? Select one or more answers.

1. It is mandatory for using Spring exception translation mechanism
2. May be improve performance when using Hibernate
3. Spring optimizes its transaction interceptor
4. Provides safeguards with Oracle and some other databases

Question 30

What data access technology is supported by the Spring framework? Select one or more answers.

1. JDBC
2. NoSQL
3. Hibernate
4. JPA

Question 31

What is not provided by the JdbcTemplate? Select a unique answer.

1. Data source access
2. Open/close data source connection
3. JDBC exception wrapping into DataAccess Exception
4. JDBC statement execution

Question 32

Using JdbcTemplate, what is the Spring provided class you will use for result set parsing and merging rows into a single object? Select a unique answer.

1. RowMapper
2. RowCallbackHandler
3. ResultSetExtractor
4. ResultSetMapper

Question 33

What configuration is supported by the LocalSessionFactoryBean which supports Hibernate 4 or higher? Select a unique answer.

1. Listing entity classes annotated with @Entity

2. Scanning a package to detect annotated entity classes (with @Entity)
3. Listing hibernate XML mapping configuration file (.hbm.xml)
4. All above

Transaction

Question 34

What is/are incorrect statements about XML declaration of the transaction manager bean? Select one or more answers.

1. The tx namespace provides JTA transaction manager declaration shortcut syntax
2. Id of the bean has to be *transactionManager*
3. Depending the application persistence technology, the HibernateTransactionManager or the DataSourceTransactionManager could be used as bean class
4. Default transaction timeout could be given

Question 35

Assuming @Transactional annotation support is enabled and the transferMoney method is called through a Spring AOP proxy, what is the behavior of the following code sample?

```java
@Transactional(propagation=Propagation.REQUIRED)
public void transferMoney(Account src, Account target, double amount) {
       add(src, -amount);
       add(src, amount);
}

@Transactional(propagation=Propagation.REQUIRES_NEW)
public void add(Account account, Double amount) {
       // IMPLEMENTATION
}
```

1. The add() method executes code in a new transaction
2. The add() method uses the transaction of the transferMoney() method
3. When calling the add() method, an exception is thrown
4. Other behavior

Question 36

Does Spring provide programmatic transaction management? Select a unique answer.

1. Yes with the TransactionTemplate class
2. Yes with the TransactionService class
3. Yes using the @Transactional bean post processor
4. No

Question 37

What is the transaction behavior of the PROPAGATION_REQUIRES_NEW mode? Select a unique answer.

1. If a transaction exists, the current method should run within this transaction. Otherwise, it should start a new transaction and run within its own transaction.
2. If a transaction is in progress, the current method should run within the nested transaction of the existing transaction. Otherwise, a new transaction has to be started and run within its own transaction.
3. The current method must start a new transaction and run within its own transaction. If there is an existing transaction in progress, it is suspended.
4. None of the above

Question 38

What is the default rollback policy in transaction management?

1. Rollback for any Exception
2. Rollback for RuntimeException
3. Rollback for checked exceptions
4. Always commit

Sping @MVC

Question 39

What could not return a Spring MVC controller? Select a single answer.

1. An absolute path to the view
2. A logical view name
3. A new JstlView
4. void
5. null value

Question 40

Where do you cannot declare Spring MVC controller? Select one or more answers.

1. In a Spring application context XML configuration file
2. Into the web.xml file of the web application
3. Into the java code by using annotations
4. Into the JSP pages

Question 41

What is the easiest method to write a unit test?

1.
   ```java
   void displayAccount(HttpServletRequest req, HttpServletResponse resp)
           throws ServletException, IOException
   ```

2.
   ```java
   void displayAccount(HttpServletRequest req, HttpSession Session)
                   throws ServletException, IOException
   ```

3.
   ```java
   @RequestMapping("/displayAccount")
   String displayAccount(@RequestParam("accountId") int id, Model model)
   ```

4.
   ```java
   @RequestMapping("/displayAccount")
   String displayAccount(@PathVariable("accountId") int id, Model model)
   ```

Spring Security

Question 42

How could you secure MVC controller with Spring Security? Select a unique answer.

1. With the @Secured annotation
2. With the @RolesAllowed annotation
3. In a XML security configuration file
4. All of the above
5. None of the above

Question 43

What are the possible mechanisms provided by Spring Security to store user details? Select one or more correct answers.

1. Database
2. JAAS
3. LDAP
4. Properties file

Question 44

What is right about Spring Security configuration and the security namespace? Select one or more correct answers.

1. The access attribute of the intercept-url tag support both EL and constants together.
2. The patterns declared into the intercept-url tag are analyzed from up to bottom. Winning is the first that matches.
3. The patterns declared into the intercept-url tag use by default the java regex syntax.
4. Security rules may apply depending request parameter

REST

Question 45

Which of the following is true regarding the below Spring controller?

```
@RestController
public class OwnerController {

    @RequestMapping(value = "/owner/{ownerId}", method = RequestMethod.POST)
    @ResponseBody
    public Owner findOwner(@PathVariable("ownerId") int ownerId) {
        return new Owner();
    }
}
```

1. RequestMethod.GET method is more accurate than POST
2. @PathVariable should be replaced with the @PathParam annotation
3. Returning the 201 HTTP status code is better
4. @ResponseBody could be removed

Question 46

Which of the following statements is true regarding the @ResponseStatus annotation?

1. @ResponseStatus is detected on nested exceptions
2. The ExceptionHandlerExceptionResolver uses the @ResponseStatus annotation to map exception to HTTP status code
3. A controller handler is annotated with the @ResponseStatus, the response status set by RedirectView takes precedence over the annotation value.
4. The @ResponseStatus annotation can go on a @RequestMapping method or a @RestController class or a business exception class.

Microservice

Question 47

Compared to monolithic application, what are the advantage(s) of microservices?

1. The base code is easy to understand
2. Imply a simple distributed system
3. Easier deployment

4. Fine-grained scaling

Question 48

What Spring Cloud provides in a microservices architecture?

1. A Service Discovery implementation
2. A server for externalized configuration
3. A Dockerfile building an image that runs any Spring Boot application
4. Netflix OSS integration for Spring Boot

Spring Boot

Question 49

What provides Spring Boot?

1. Support for Jetty and Undertow as embedded containers
2. Java code generation
3. Auto-configuration of the Spring Framework and third libraries
4. Convenient dependency descriptors to load transitive dependencies
5. Support both Java-based and YAML for Spring application context configuration

Question 50

What is the name of the default environment configuration file of Spring Boot?

1. configuration.spring
2. configuration.yml
3. configuration.xml
4. application.properties
5. application.json