**SpringBoot and MicroServices TEST**

**Objective: 45 Minutes**

**Q1. X and Y are working in a team as a developer in a project. X is working on Microservice A and Y is working on Microservice B. The Microserice A has to call the Microservice B. The project is expecting to use declarative approach to call Microservice B rather than programatic approach. Which would be the right choice to setup inter-microservice communication in this scenario?**

1. Feign Client
2. Rest Template
3. Anyone can be used
4. None of above

**Q2.** **Which of the following is false?**

1. A BeanFactory pretty much just instantiates and configures beans.
2. An ApplicationContext also instantiates bean,and it provides the supporting infrastructure to enable lots of enterprise-specific features such as transactions and AOP,message resource handling (for use in internationalization), event propagation.

Application-layer specific contexts such as the WebApplicationContext use in web application

1. non of the above

**Q3.** **package com;**

**public class TestBean {**

**private int year;**

**private String happy;**

**public TestBean( String happy,int year) {**

**this.year = year;**

**this.happy = happy;**

**}**

**}**

**What is the correct way to write constructor injection?**

**A** .<bean name="testClass" class="com.TestBean">

<constructor-arg type="java.lang.String" value="Happy new year"/>

<constructor-arg type="int" value="2015"/>

</bean>

**B.** <bean name="testClass" class="com.TestBean">

<constructor-arg index="0" value="Happy new year"/>

<constructor-arg index="1" value="2015"/>

</bean>

**C.** <bean name="testClass" class="com.bullraider.TestBean">

<constructor-arg index="1" value="Happy new year"/>

<constructor-arg index="0" value="2015"/>

</bean>

**D.**<bean name="testClass" class="com.TestBean">

<constructor-arg type="int" value="2015"/>

<constructor-arg type="java.lang.String" value="Happy new year"/>

</bean>

**E.** All the above

**Q4. @RequestParam is useful for binding query parameters to method parameters where the names don't match.**

1. true
2. false

**Q5. What is true about @Autowired annotation?**

A. The @Autowired annotation can be used to autowire bean on the setter method.

B. This annotation provides more fine-grained control over where and how autowiring should be accomplished.

C. The @Autowired annotation can be used to autowire bean on the methods with arbitrary names and/or multiple arguments.

D. All of above.

**Q6. How to use ref keyword in beans.xml?**

A. Using setter method only.

B. Using constructor argument only.

C. Using setter method and constructor argument both.

D. None of the above.

**Q7. Which of the following is correct about dependency injection?**

A. It helps in decoupling application objects from each other.

B. It helps in deciding the dependencies of objects.

C. It stores objects states in database.

D. It stores object states in file system.

**Q8. \_\_\_\_\_\_\_\_\_\_\_ annotation has been added in the main spring boot configuration class to make the service as a Discoverable client. Fill in the blanks?**

1. @EnableDiscoveryService
2. @DiscoveryClient
3. @EurekaClient
4. @EnableDiscoveryClient

**Q9. Amit wants to register student-service on Eureka Server. Which of the following is valid application.properties file to register student-service on Eureka server?  
Note : Eureka Server is running on port no 8761.**

1. spring.application.name= student-service  
   server.port=8098  
   eureka.client.serviceUrl.defaultZone = http://localhost:8761/eureka/
2. spring.application.name= student-service  
   server.port=8098  
   eureka.client.serviceUrl = http://localhost:8761/eureka/
3. spring.service.name= student-service  
   server.port=8098  
   eureka.client.serviceUrl.defaultZone = http://localhost:8761/eureka/
4. None of the above

**Q10. When we use microservices?**

1. When we need to create large, enterprise-level applications that are subject to changes on a frequent basis.
2. When we work with ephemeral nano technology
3. When we create applications specifically for scientific test equipment
4. All the above

**Q11. XYZ bank has setup all their services in microservice architectural pattern. Couples service have been stopped servicing the requests which is raising some exception and enviornment is getting unsatable due to single point of failure. What solution do you suggest?**

1. We can set up one micorservice which will deal with execption and we can call that microservice.
2. We can redirect the traffic to fallback path and can monitor the microservices.
3. We can catch the exception, FastFailException, that is thrown by the circuit breaker when it is open and call a fallback method.
4. None of the above

**Q12. What will happen if some microservices fail or any exception is thrown by some services? Then this exception is propagated to upstream services and finally comes to the end-user. Which of the following pattern prevents failure cascading and gives a default behavior when services fail?**

1. Circuit Breaker
2. Service Discovery
3. Api gateway
4. None of the above

**Q13. Akash is an application developer and he currently working with a RESTful web service called EmployeeService which is developed using Spring Boot Framework. Help him to identify the dependency in his pom.xml.**

1. Akash needs to include the below dependency in his pom.xml application

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

1. Akash needs to include the below dependency in his pom.xml application

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

1. Akash needs to include the below dependency in his pom.xml application

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

1. Akash needs to include the below dependency in his pom.xml application

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

**Q14. If a bean can be created any number of times, scope is**

[**A** session](javascript:void(0);)

[**B** global-session](javascript:void(0);)

**C** prototype

[**D** request](javascript:void(0);)

**Q15.  A bean can be requested by**  
a) getBean method  
b) reference from another bean using autowiring, property etc  
c) all of the mentioned  
d) none of the mentioned

**Q16. Which one is the default scope of the beans?**  
a) Prototype  
b) Session  
c) Request  
d) Singleton

**Q17. The @RequestMapping annotation can be applied to the class level only.**a) True  
b) False

**Q18. Which of the following is correct assertion about spring?**

[**A**  Spring enables developers to develop enterprise-class applications using POJOs.](javascript:void(0);)

[**B** Spring is organized in a modular fashion.](javascript:void(0);)

[**C** Testing an application written with spring is simple because environment-dependent code is moved into this framework.](javascript:void(0);)

**D** All of above.

**Q19. X and Y are working in a team as a developer in a project. X is working on Currency Conversion Service and Y is working on CurrencyExchangeService. The CurrencyConversionService has to call the CurrencyExchangeService. Which of the following is valid proxy declaration for feign client?  
Note: CurrencyExchangeService is running on port 8000 and it is not discovered/registered on eureka server**

1. import org.springframework.cloud.openfeign.FeignClient;  
     
   @FeignClient  
   public interface CurrencyExchangeServiceProxy {  
   }
2. import org.springframework.cloud.openfeign.FeignClient;  
     
   @FeignClient(name="CurrencyExchangeService")  
   public interface CurrencyExchangeServiceProxy {  
   }
3. import org.springframework.cloud.openfeign.FeignClient;  
     
   @FeignClient(name="CurrencyExchangeService",url="localhost:8000")  
   public interface CurrencyExchangeServiceProxy {  
   }
4. None of the above

**Q20. Maria is working on microservice architecture and she wants to store all configuration file on cloud. Which of the following code is valid to enable cloud config server?**

1. import org.springframework.boot.SpringApplication;  
   import org.springframework.boot.autoconfigure.SpringBootApplication;  
   import org.springframework.cloud.config.server.EnableCloudConfigServer;  
     
   @EnableCloudConfigServer  
   @SpringBootApplication  
   public class SpringCloudConfigServerApplication {  
   }
2. import org.springframework.boot.SpringApplication;  
   import org.springframework.boot.autoconfigure.SpringBootApplication;  
   import org.springframework.cloud.config.server.EnableCloudConfig;  
     
   @EnableCloudConfig  
   @SpringBootApplication  
   public class SpringCloudConfigServerApplication {  
   }
3. import org.springframework.boot.SpringApplication;  
   import org.springframework.boot.autoconfigure.SpringBootApplication;  
   import org.springframework.cloud.config.server.EnableConfigServer;  
     
   @EnableConfigServer  
   @SpringBootApplication  
   public class SpringCloudConfigServerApplication {  
   }
4. All the above