IT Skill Test GIC Myanmar

Duration: 60 Minutes Total Questions: 15

---

1. You are developing a Spring Boot application and need to configure a bean. Which of the following annotations is used to define a bean in a configuration class?

   A.@Component

   B.@Bean

   C.@Autowired

   D.@Service

2. Consider the following Spring Boot application:

```java
@SpringBootApplication
public class MyApplication {
    public static void main(String[] args) {
        SpringApplication.run(MyApplication.class, args);
    }
}

@RestController
class MyController {
    @Autowired
    private MyService myService;

    @GetMapping("/hello")
    public String hello() {
        return myService.getMessage();
    }
}
```

```
interface MyService {
    String getMessage();
}

@Service
class MyServiceImpl implements MyService {
    public String getMessage() {
        return "Hello, World!";
    }
}
```

When you run this application and access the '/hello' endpoint, what will be the output?

A. The application will fail to start due to a missing bean definition

B. The application will start, but accessing '/hello' will result in a NullPointerException

C. The application will start and accessing '/hello' will return 'Hello, World!'

D. The application will start, but accessing '/hello' will result in a 404 Not Found error

3. You're debugging a Spring Boot application and notice that a particular bean is not being autowired correctly. Which of the following could be a potential cause of this issue?

A. The bean class is not annotated with @Component or a stereotype annotation

B. The @Autowired annotation is missing from the field or constructor

C. The bean is not in the component scan path

D. All of the above

4. In a Spring Boot application, you want to configure a datasource programmatically. Which of the following code snippets correctly creates a DataSource bean?

A.
```java
@Bean
public DataSource dataSource() {
    return new DriverManagerDataSource();
}
```

B.
```java
@Bean
public DataSource dataSource() {
    return DataSource.builder()
        .url("jdbc:mysql://localhost/mydb")
        .username("user")
        .password("pass")
        .build();
}
```

C.
```java
@Bean
public DataSource dataSource() {
    return DataSourceBuilder.create()
        .url("jdbc:mysql://localhost/mydb")
        .username("user")
        .password("pass")
        .build();
}
```

D.
```java
@Bean
public DataSource dataSource() {
    DataSource ds = new DataSource();
    ds.setUrl("jdbc:mysql://localhost/mydb");
```

```
    ds.setUsername("user");
    ds.setPassword("pass");
    return ds;
}
```

5. You are developing a Spring Boot application and need to handle exceptions globally. Which of the following annotations should you use on a class to achieve this?

A.@ExceptionHandler

B.@ControllerAdvice

C.@GlobalExceptionHandler

D.@ErrorController

6. In a Spring Boot application, you want to use constructor injection for a service. Which of the following code snippets correctly implements constructor injection?

A.```java
```
@Service
public class MyService {
    private final Repository repository;

    @Autowired
    public MyService(Repository repository) {
        this.repository = repository;
    }
}
```
```

B.```java
```
@Service
public class MyService {
    @Autowired
    private Repository repository;
```
```

```
    public MyService() {}
}
```

C.```java
@Service
public class MyService {
    private Repository repository;

    public MyService(Repository repository) {
        this.repository = repository;
    }
}
```

D.```java
@Service
public class MyService {
    private Repository repository;

    @Inject
    public void setRepository(Repository repository) {
        this.repository = repository;
    }
}
```

7. You are working on a Spring Boot application and need to configure it to use a specific port. Which of the following is the correct way to set the server port in the application.properties file?

A.port=8080

B.server.port=8080

C.application.port=8080

D.spring.port=8080

8. You are developing a Spring Boot application and need to implement a custom health check. Which of the following correctly implements a custom health indicator?

A.
```java
@Component
public class CustomHealthIndicator implements HealthIndicator {
    @Override
    public Health health() {
        int errorCode = check();
        if (errorCode != 0) {
            return Health.down().withDetail("Error Code", errorCode).build();
        }
        return Health.up().build();
    }

    private int check() {
        // perform health check
        return 0;
    }
}
```

B.
```java
@Component
public class CustomHealthCheck {
    @HealthCheck
    public boolean performHealthCheck() {
        // perform health check
        return true;
    }
}
```

C.
```java
@Component
public class CustomHealthIndicator extends Health {
    @Override
```

```
    public Health getHealth() {
        // perform health check
        return Health.up().build();
    }
}
```

D.
```java
@Component
public class CustomHealthIndicator {
    @Bean
    public Health health() {
        // perform health check
        return Health.up().build();
    }
}
```

9. You're debugging a Spring Boot application and notice that a particular bean is not being created. What's the most likely cause?

A.The class is not in the base package or its sub-packages

B.The class is missing a no-args constructor

C.The class is not annotated with @Component or a stereotype annotation

D.The application.properties file is missing

10. You're working on a Spring Boot application and need to run some code exactly once, after the application context has been initialized. Which of the following approaches would you use?

A.Implement the ApplicationListener interface

B.Use the @PostConstruct annotation on a method

C.Implement the CommandLineRunner interface

D.Use the @Scheduled annotation with a fixed delay

11. You're developing a RESTful API using Spring Boot. You want to handle exceptions globally and return appropriate HTTP status codes. Which of the following is the best approach?

    A.Use try-catch blocks in each controller method

    B.Implement a custom error page

    C.Use @ExceptionHandler methods in each controller

    D.Create a global @ControllerAdvice class with @ExceptionHandler methods

12. Consider the following Spring Boot configuration class:

    ```java
    @Configuration
    public class MyConfig {
        @Bean
        public MyBean myBean() {
            return new MyBean();
        }

        @Bean
        public AnotherBean anotherBean(MyBean myBean) {
            return new AnotherBean(myBean);
        }
    }
    ```

    What does this configuration demonstrate?

    A.Constructor injection

    B.Setter injection

    C.Field injection

    D.Method injection

13. You're working on a Spring Boot application and need to run some initialization code when the application starts, but only if a specific profile is active. Which of the following is the best approach?

    A.Use @Profile annotation on a @Configuration class

    B.Use @ConditionalOnProperty annotation

    C.Implement the ApplicationListener interface and check the active profile

    D.Use @Value annotation to inject the active profile and check it in a @PostConstruct method

14. In a Spring Boot application, you need to configure multiple datasources. Which of the following is the correct approach?

    A.Define multiple DataSource beans in a @Configuration class

    B.Use @Primary annotation on one DataSource bean

    C.Configure multiple datasources in application.properties file

    D.Both A and B

15. You're developing a Spring Boot application and want to use aspect-oriented programming (AOP) to log method execution times. Which of the following is the correct way to enable AOP in your application?

    A.Add @EnableAspectJAutoProxy to the main application class

    B.Include spring-boot-starter-aop dependency in your pom.xml

    C.Implement the Aspect interface in your aspect class

    D.Use @Configuration annotation on your aspect class