IT Skill Test GIC Myanmar

Duration: 60 Minutes Total Questions: 15

---

1. You are working on a Spring Boot application that uses the Repository Pattern. You notice that one of your repository interfaces is not being recognized by Spring Data JPA. Which of the following is the most likely cause of this issue?

   A.The repository interface is not extending JpaRepository

   B.The @Repository annotation is missing from the interface

   C.The entity class is not properly annotated with @Entity

   D.The repository interface is not in the same package as the main application class

2. You are developing a RESTful API using Spring Boot. You have created a controller class, but the endpoints are not accessible. Which of the following annotations is missing from your controller class?

   A.@Controller

   B.@Service

   C.@RestController

   D.@Repository

3. Consider the following code snippet from a service class:

   ```java
   @Service
   public class UserService {
       @Autowired
       private UserRepository userRepository;
   ```

```
    public User findById(Long id) {
        return userRepository.findById(id).orElse(null);
    }
}
```

What is the potential issue with this implementation?

A.The @Service annotation is unnecessary

B.The method should return Optional<User> instead of User

C.The @Autowired annotation should be on a constructor instead of a field

D.The userRepository.findById(id) method doesn't exist

4. You are implementing a RESTful API endpoint to create a new user. Which HTTP method should you use for this operation?

A.GET

B.POST

C.PUT

D.DELETE

5. In a Spring Boot application, you have the following repository interface:

```java
public interface ProductRepository extends JpaRepository<Product, Long> {
    List<Product> findByPriceGreaterThan(BigDecimal price);
}
```

What will happen when you try to use this method?

A.It will cause a compilation error

B.It will throw a runtime exception

C.It will work correctly without any additional implementation

D. It requires a custom implementation in a separate class

6. You are debugging a Spring Boot application and notice that a particular service method is being called multiple times unnecessarily. Which Spring annotation can you use to cache the results of this method?

A.@Cacheable

B.@Transactional

C.@Async

D.@Scheduled

7. You are working on a Spring Boot application that uses JPA for data persistence. You notice that database operations are slow, and you want to optimize them. Which of the following approaches would be most effective?

A.Use @Transactional on all repository methods

B.Implement batch processing for bulk operations

C.Always use eager fetching for all entity relationships

D.Use native SQL queries instead of JPQL for all operations

8. You are implementing a service layer in your Spring Boot application. Which of the following is a best practice for service layer implementation?

A.Services should directly access the database without using repositories

B.Services should contain the business logic of the application

C.Services should be stateful and maintain user session information

D.Services should handle HTTP request and response formatting

9. You are working on a Spring Boot application and need to implement a custom query that cannot be expressed using Spring Data JPA's method naming conventions. Which of the following is the most appropriate way to implement this query?

A. Write a native SQL query in the repository interface using @Query

B. Implement the query in the service layer using JDBC

C. Use the @Query annotation with JPQL in the repository interface

D. Create a new DAO class that extends JdbcTemplate

10. You are working on a Spring Boot application that uses the Repository pattern. You notice that one of your repository interfaces is not being recognized by Spring Data JPA. Which of the following is the most likely cause of this issue?

A. The repository interface is not extending JpaRepository or CrudRepository

B. The @Repository annotation is missing from the interface

C. The entity class is not properly annotated with @Entity

D. The repository interface is not in the same package as the main application class

11. You are implementing a REST API using Spring Boot. You have created a controller class, but the endpoints are not accessible. Given the following code snippet, what is the issue?

```
public class UserController {
    @Autowired
    private UserService userService;

    @GetMapping("/users")
    public List<User> getAllUsers() {
        return userService.getAllUsers();
    }
}
```

A. The @RestController annotation is missing from the class

B. The @Service annotation is missing from the UserService class

C. The @Autowired annotation should be replaced with constructor injection

D. The @GetMapping annotation is incorrect for RESTful endpoints

12. In a Spring Boot application, you're implementing a service layer. Which of the following code snippets represents the best practice for dependency injection in a service class?

A.
```
@Autowired
private UserRepository userRepository;
```

B.
```
private UserRepository userRepository;

@Autowired
public UserService(UserRepository userRepository) {
    this.userRepository = userRepository;
}
```

C.
```
private static UserRepository userRepository;

public UserService() {
    userRepository = new UserRepository();
}
```

D.
```
public UserService() {
    this.userRepository = BeanFactory.getBean(UserRepository.class);
}
```

13. You are debugging a Spring Boot application and notice that a particular service method is not being executed as expected. Upon inspection, you find the following code:

```
@Service
public class OrderService {
    @Transactional(readOnly = true)
    public void createOrder(Order order) {
        // ... implementation
    }
}
```

What is the likely issue with this code?

A.The @Service annotation is incorrect for service classes

B.The method name doesn't follow Spring naming conventions

C.The @Transactional annotation is set to readOnly for a write operation

D.The method should return an Order object instead of void

14. In a Spring Boot application using JPA, you need to implement a method in your repository to find users by their email domain. Which of the following method signatures in the UserRepository interface would correctly implement this using a derived query method?

A.List<User> findByEmailContaining(String domain);

B.List<User> findByEmailEndingWith(String domain);

C.List<User> findByEmailDomain(String domain);

D.List<User> findUsersByEmailDomain(String domain);

15. In a Spring Boot application, you need to implement a service method that should only commit changes to the database if all operations within the method succeed. Which of the following annotations should you use on the method to achieve this behavior?

A.@Transactional(propagation = Propagation.REQUIRED)

B.@Transactional(isolation = Isolation.SERIALIZABLE)

C.@Transactional(readOnly = false)

D.@Transactional(noRollbackFor = Exception.class)