1. You are to define a bean definition “getServiceBean(), in a Java configuration class for EmployeeServiceImpl with a name of “emplService” and a public initializer method of “init”. Which is a correct approach to doing this? 1 / 1 point

- [x] @Bean(name=”emplService”, init-method=”init”)

Correct
Correct; the name element names the bean emplService and executes init() as part of its lifecycle

- [x] @Bean(name=”emplService”) and annotate method “init” in EmployeeServiceImpl with @PostConstruct

Correct
Correct; the name element names the bean emplService and executes init() as part of its lifecycle because annotations are “active” and @PostConstruct will fire at the appropriate part of the bean lifecycle

- [x] All of the above

Correct
Correct

- [x] @Bean(name=”emplService”) and execute the init() method inside the Bean definition method itself before returning the service bean

Correct
Correct; the name element names the bean emplService and the execution of init() is explicitly executed within the bean definition method prior to returning the bean

2. Using Annotations, how could you inject a bean that implements the Interface EmployeeDao, whose id is “emplDao” into a Spring Managed bean of “employeeService”? Assume that there are multiple beans that implement EmployeeDao in your application and you are annotating the property employeeDaoImpl (field). 1 / 1 point

- ( ) @Named(“emplDao”)
- ( ) @Value(“emplDao”)
- (•) @Resource(name=“emplDao”)
- ( ) @Autowired (name=“emplDao”)

Correct
Correct; JSR250 @Resource wires beans up by name or id

3. You are writing a JUnit test class for your application using a Java Configuration class AppConfig, It is also to have an Initializer (the class MyInitiaizer) where you have tailored System Properties for use in the test. Which is the Annotation approach that you will take 1 / 1 point

- ( ) ExtendWith(SpringExtension.class) and @ContextConfiguration(locations=AppConfig.**class, initializer=MyInitializer.class**)
- ( ) ExtendWith(SpringExtension.class) and @ContextConfiguration(classes=AppConfig.**class, initializer=”MyInitializer”**)
- ( ) @ExtendWith(SpringExtension.class) and @ContextConfiguration(classes=AppConfig.**class**)
- (•) @ExtendWith(SpringExtension.class) and @ContextConfiguration(classes=AppConfig.**class, initializer=MyInitializer.class**)

Correct
Correct; the classes attribute classes points to the configuration class and initializer to a registered Initializer

4. Which of the following is false about JSR 250 @Resource annotation? 1 / 1 point

- (●) Can be used on constructors
- ( ) Is not a Spring Annotation
- ( ) Can only be used on setters
- ( ) Does the same as @Autowired as it can try to resolve a bean by-type

Correct
False; It cannot be used with a constructor

5. We mentioned that after setter injection in a bean lifecycle, we can define an initializer by implementing what Interface 1 / 1 point

- ( ) DisposableBean
- ( ) ApplicationContextAware
- (●) InitializingBean
- ( ) BeanPostProcessor

Correct
Correct; this has a method afterPropertiesSet that the framework will trigger after setter injection has completed

6. Which of these annotations are JSR330 1 / 1 point

- ( ) @Resource
- ( ) @Override
- ( ) @Autowired
- (●) @Inject

Correct
Correct; this is the equivalent of Autowired from JSR330

7. What were the motivations behind Java Configurations; pick all that apply 1 / 1 point

- [x] Get rid of XML

Correct
Correct; One less editor unfriendly dependency

- [x] Type Safety

Correct
Correct; Java is type safe

- [x] Externalize the configuration

Correct
Correct; Configuration classes are external to the domain code, but we can still use annotations in them if we want, they are still active

- [x] Compile time checking

Correct
Correct; as its java it is compiled before anything is run

8. What annotation can we use for simple value injection

1 / 1 point

- ○ @Configuration
- ◉ @Value
- ○ @Scope
- ○ @Component

Correct
Correct; @Value(“..”) over a field, constructor argument, setter

9. What tag from the context namespace do you use to register the BeanPostProcessors that scan for annotations or spring managed bean

1 / 1 point

- ☐ property
- ☑ component-scan

Correct
Correct this tags scans for @Component, @Named beans or via a filter

- ☐ bean
- ☑ annotation-config

Correct
Correct this tag registered all BeanPostProcessors for the common annotations

10. Which annotation do you use @Qualifier with

1 / 1 point

- ◉ @Autowired
- ○ @Inject
- ○ @PostConstruct
- ○ @Resource

Correct
Correct; @Autowired @Qualifier(id of a bean or its Qualifier value when Qualifier is used at the class level)