Spring Framework

Q1. How filters are used in Spring Web?

- ✓ Filters are called before a request hits the DispatcherServlet. They allow for interception-style, chained processing of web requests for security, timeouts, and other purposes.
- ☐ Filters are used with a checksum algorithm that will filter invalid bytes out of a byte stream request body and allow for processing of HTTP requests from the DispatcherRequestServlet.
- ☐ Filters are used with a checksum algorithm that will filter invalid bytes out of an octet stream a multipart upload and allow for chained processing of WebDispatcherServlet requests.
- ☐ Filters are used to validate request parameters out of the byte stream request body and allow for processing of requests from the DispatcherRequestServlet.

HandlerInterceptors vs. Filters in Spring MVC (https://www.baeldung.com/spring-mvc-handlerinterceptor-vs-filter). Also there is no such thing as DispatcherRequestServlet in Spring.

Q2. How is a resource defined in the context of a REST service?

- ☐ A resource is the actual String literal that composes a URI that is accessed on a RESTful web service.
- ✓ It is an abstract concept that represents a typed object, data, relationships, and a set of methods that operate on it that is accessed via a URI.
- ☐ A REST service has a pool of resources composed of allocations of memory that allow a request to be processed.
- ☐ A resource for a REST service is an explicit allocation of a thread or CPU cycles to allow a request to be processed.

Rest Service in Spring (https://spring.io/guides/tutorials/rest/)

Q3. Which of these is a valid Advice annotation?

- ☐ @AfterError
- ✓ @AfterReturning
- ☐ @AfterException
- ☐ @AfterExecution

Spring Advice Type Annotation (https://www.baeldung.com/spring-aop-advice-tutorial)

Q4. What does a ViewResolver do?

- ☐ It supports internationalization of web applications by detecting a user's locale.
- ✓ It generates a view by mapping a logical view name returned by a controller method to a view technology.
- ☐ It creates a unique view determined by the uers's browser type,supporting cross-browser compatibility.
- ☐ It maps custom parameters to SQL views in the database, allowing for dynamic content to be created in the response.

View Resolution in Spring MVC (https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-servlet/viewresolver.html)

Q5. How are Spring Data repositories implemented by Spring at runtime?

- ☐ Spring automatically generated code for you based on your YAML config that defined a MethodInterceptor chain that intercept calls to the instance and computed SQL on the fly.
- ☑ A JDK proxy instance is created, which backs the repository interface, and a MethodInterceptor intercepts calls to the instance and routes as required.
- ☐ The Spring JDK proxy creates a separate runtime process that acts as an intermediary between the database and the Web server, and intercepts calls to the instance and handles requests.
- ☐ Spring automatically generated code for you based on your XML config files that define a SpringMethodAutoGeneration factory that intercepts calls to the instance and creates dynamic method that computer SQL on the fly.

Explained QnA (https://stackoverflow.com/questions/38509882/how-are-spring-data-repositories-actually-implemented)

Q6. What is SpEL and how is it used in Spring?

- ☐ SpEL(Spring Expression Language) runs in the JVM and can act as a drop-in replacement for Groovy or other languages.
- ☑ SpEL(Spring Expression Language) supports boolean and relational operators and regular expressions, and is used for querying a graph of objects at runtime.
- ☐ SpEL(Spring Expression Language) allows you to build, configure,and execute tasks such as building artifacts and downloading object dependencies.
- ☐ SpEL(Spring Expression Language) natively transpiles one JVM language to another, allowing for greater flexibility.

Spring Expression Documentation (https://docs.spring.io/spring-framework/docs/3.0.x/reference/expressions.html)

Q7. The process of linking aspects with other objects to create an advised object is called

- ☐ dynamic chaining
- ☐ banding
- ☑ weaving
- ☐ interleaving

Q8. How are JDK Dynamic proxies and CGLIB proxies used in Spring?

- ☑ JDK Dynamic proxy can proxy only interface, so it is used if the target implements at least one interface. A CGLIB proxy can create a proxy by subclassing and is used if the target does not implement an interface.
- ☐ Only JDK Dynamic proxies are used in the Spring Bean Lifecycle. CGLIB proxies are used only for integrating with other frameworks.
- ☐ Only CGLIB proxies are used in the Spring Bean Lifecycle. JDK Dynamic proxies are used only for integrating with other frameworks.

- ☐ JDK Dynamic proxy can only using an abstract class extended by a target. A CGLIB proxy can create a proxy through bytecode interweaving and is used if the target does not extend an abstract class.

Q9. Which of these is not a valid method on the JoinPoint interface?

- ☐ getArgs()
- ✓ getExceptions()
- ☐ getSignature()
- ☐ getTarget()

Q10. In what order do the @PostConstruct annotated method, the init-method parameter method on beans and the afterPropertiesSet() method execute?

- ☐ 1. afterPropertiesSet() 2. init-method 3. @PostConstruct
- ✓ 1. @PostConstruct 2. afterPropertiesSet() 3. init-method
- ☐ 1. init-method 2. afterPropertiesSet() 3. @PostConstruct
- ☐ You cannot use these methods together-you must choose only one.

Q11. What is the function of the `@Transactional` annotation at the class level?

- ☐ It's a transaction attribute configured by `spring.security.transactions.xml` config file that uses Spring's transaction implementation and validation code.
- ☐ It's a transaction that must actively validate by the bytecode of a transaction using Spring's `TransactionBytecodeValidator` class. Default Transaction behavior rolls back on validation exception but commits on proper validation
- ✓ It creates a proxy that implements the same interface(s) as the annotated class, allowing Spring to inject behaviors before, after, or around method calls into the object being proxied.
- ☐ It's a transaction that must be actively validated by Spring's `TransactionValidator` class using Spring's transaction validation code. Default Transaction behavior rolls back on validation exception.

Q12. Which is a valid example of the output from this code (ignoring logging statements) ?

```
@SpringBootApplication
public class App {
     public static void main(String args[]) {
          SpringApplication.run(App.class, args);
          System.out.println("startup");
     }
}

public class Print implements InitializingBean {
     @Override
     public void afterPropertiesSet() throws Exception {
          System.out.println("init");
     }
}
```

- ☐ Nothing will print
- ☐ startup init
- ☐ init
- ✓ startup

Explanation: SpringApplication.run method returns the created Context, so main method will continue running and print "startup". Class Print is not a Spring Bean, because it is not annotated with @Component, so it will not be initialized.

Q13. Which println statement would you remove to stop this code throwing a null pointer exception?

```
@Component
public class Test implements InitializingBean {
     @Autowired
     ApplicationContext context;
     @Autowired
     static SimpleDateFormat formatter;

     @Override
     public void afterPropertiesSet() throws Exception {
          System.out.println(context.containsBean("formatter") + " ");
          System.out.println(context.getBean("formatter").getClass());
          System.out.println(formatter.getClass());
          System.out.println(context.getClass());
     }
}

@Configuration
class TestConfig {
     @Bean
     public SimpleDateFormat formatter() {
          return new SimpleDateFormat();
     }
}
```

- ✓ formatter.getClass()
- ☐ context.getClass()
- ☐ context.getBean("formatter").getClass()
- ☐ context.containsBean("formatter")

Q14. What is the root interface for accessing a Spring bean container?

- ☐ SpringInitContainer
- ☐ ResourceLoader
- ☐ ApplicationEventPublisher
- ✓ BeanFactory

Q15. Which annotation can be used within Spring Security to apply method level security?

- ✓ @Secured
- ☐ @RequiresRole
- ☐ @RestrictedTo
- ☐ @SecurePath

Q16. What is the result of calling the map controller method using the HTTP request GET localhost:8080/map?foo=foo&bar=bar ?

```
@RestController
public class SampleController {


    @RequestMapping("/map")
    public String map(@RequestParam("bar") String foo, @RequestParam("foo") String bar) {
        return bar + foo;
    }
}
```

- ☐ An InvalidParameterNameMappingException is thrown at runtime.
- ☐ barfoo
- ✓ foobar
- ☐ A status code of 400 Bad Request is returned.

Q17. What is the purpose of the @Lazy annotation and why would you use it?

- ☐ It prevents a bean from being created and injected until you run a specific CLI command. It reduces complexity in the application.
- ☐ It can be applied to a bean so that the bean is not persisted in the database. It reduces the number of database operations.
- ✓ It can be applied to a bean so that the bean is not created and injected until it is needed. It can help speed up startup time for your application.
- ☐ It prevents a bean from being created and injected until it receives a HTTP request to a callback hook. It reduces disk footprint.

Q18. What is dependency injection?

- ✓ a method by which objects define dependencies they need as abstractions that allows the framework to instantiate and configure them in a central location at runtime.
- ☐ a paradigm where dependent code is injected into the bytecode of a Java application on a remote server.
- ☐ a way of injecting remote dependencies into a pre-packaged JAR file from the file system.
- ☐ a way of injecting remote dependencies into a pre-packaged WAR file from the file system.

Q19. What is a RESTful web service?

- ☐ Reactive Enterprise Status Transfer is a web service comprising a set of guidelines and technical constraints for web services that monitor and alert of a set of mission-critical resources.
- ✓ Representational State Transfer an architectural style for creating web services that includes client-server architecture, statelessness, cacheability, a layered system, and a uniform interface.
- ☐ Redundant Enumerated State Transfer is a web service that provides redundancy in the case of failure and outlines technical constraints for web services that have access to resources.
- ☐ Reactive Enterprise State Transfer is a web service comprising a set of guidelines and technical constraints for web services y that have access to resources and are async in nature.

Q20. What happens when a class is annotated with the @Controller annotation?

- [✓] A controller bean definition is defined in the servlet's WebApplicationContext. The class is marked as a web component, and you can map requests to controller methods.
- [ ] A controller bean definition is defined in the Web Context, and Web Servlet is marked as a component that reads mapped controller requests from an XML config file.
- [ ] A controller bean definition is defined in the Tomcat Context, and the Controller Servlet is marked as a web component that reads mapped controller requests from a YAML config file.
- [ ] A controller bean definition is defined in the Servlet Context, and the Controller Servlet is marked as a component that reads mapped controller requests from an XML config file.

Q21. Which property can be used to change the port of a Spring application?

- [ ] Port
- [ ] spring.port
- [ ] spring.settings.port
- [✓] server.port

Q22. What is the purpose of the @ResponseBody annotation?

- [ ] to validate the char array contained in a response to ensure that it is a valid character encoding
- [ ] to generate a local byte stream from the body of a response that allows a request to be scanned for security risks
- [✓] to indicate whether a handler method return value should be bound to the web response body in servlet environments
- [ ] to ensure a ThreadLocal byte stream that allows a response to be encoded for reading directly into a socket stream

Q23. How are mocking frameworks such as Mockito used with Spring?

- [ ] Mockito will spin up a mock web service hosted in a Docker container that can respond to HTTP requests to mock out third-party APIs.
- [✓] Mockito can use annotations or factory methods to create mock objects that may be injected into tests in place of dependencies. The behavior of mocks can be explicitly configured.
- [ ] Mockito will create mock users to repeatedly perform requests against your Spring application to test your app's ability to take load.
- [ ] Mockito will spin up a mock web service hosted in a Docker container that can respond to RPC calls to mock out remote resources.

Q24. What is the name of the central servlet that dispatches requests to controllers?

- [ ] DispatchingDelegatorServlet
- [✓] DispatcherServlet
- [ ] Router
- [ ] FrontControllerServlet

Q25. What is the purpose of the Spring IoC (Inversion of Control) container?

- ☑ It instantiates and configures objects, supplied at runtime, to classes that define them as a dependency.
- ☐ It allows the front-end code to manage the ResponseBody objects provided by a back-end REST API.
- ☐ It allows a database to define business objects via a shared schema at compile time.
- ☐ It facilitates a remote server to configure a local application.

Q26. What is component scanning?

- ☑ feature that scans packages for classes with specific annotations and, when found, creates their bean definitions within the IoC container
- ☐ paradigm where bytecode is actively scanned to identify additional optimizations to inject into components in the application context
- ☐ a method by which cloud repositories are scanned to identify components for injection into an IoC container
- ☐ a method by which binary data in a database is searched to identify components for injection into the IoC container

Q27. What does @SpringBootApplication do?

- ☐ This annotation takes the String literal passed into the annotation as a parameter and automatically generates all the code for your application as per the passed in template parameter.
- ☑ This compound annotation applies the @Bootable, @Springify, and @StandardConfig annotations that launch a CLI tool after launching the Spring Boot WAR file that will guide you through a series of prompts to set up your app.
- ☐ This annotation scans the provided spring-boot-config-construction.yaml file in your root directory and automatically generates all the code for your application as defined in the YAML file.

Q28. How does Spring Data facilitate queries against a datastore?

- ☐ Queries are explicitly coded in repository implementations using the Spring Data CriteriaBuilder.
- ☐ Query metadata is stored in the underlying datastore and retrieved at runtime per repository.
- ☑ Queries are derived from the signatures of methods on a Spring Data repository that contain keywords in their name for constructing the query logic.
- ☐ A spring-data-queries.xml file contains queries within entity tags that specify the query logic for each repository.

Q29. How does Spring generate bean names for classes annotated with @Component that do not specify a name?

- ☑ It uses the short name of the class with the first letter in lowercase.
- ☐ It uses the short name of the class.
- ☐ It uses the short name of the class in uppercase.
- ☐ It uses the canonical name of the class in lowercase.

Q30. What is the delegating filter proxy?

- ☐ It's the servlet filter chain proxy that handles all requests to the route defined in spring.security.xml. All calls to the filter proxy are forwarded to the ErrorDispatcherServlet.
- ☐ It's the servlet filter chain that handles requests to the route defined in spring.security.factories. All calls to the filter proxy y are forwarded to the ErrorServlet.

- [✓] It's the servlet filter proxy delegating to a filter bean specified in web.xml. All calls to the filter proxy will be delegated to that servlet filter bean.
- [ ] It's the web servlet daemon filter proxy that delegates to a bean specified in spring.security.factories. All calls to the filter proxy that do not contain a proper route will return an error.

Overview and Need for DelegatingFilterProxy in Spring (https://www.baeldung.com/spring-delegating-filter-proxy)

Q31. What value does Spring Boot Actuator provide?

- [✓] It helps monitor and manage a Spring Boot application by providing endpoints such as health checks, auditing, metrics gathering, and HTTP tracing.
- [ ] It provides out-of-the-box functionality that integrates with third-party metrics platforms to automatically scale up and down the number of threads in threadpools.
- [ ] It's a CLI that allows you to modify the configuration of a running Spring Boot application without the need for restarting or downtime.
- [ ] It provides out-of-the-box functionality that integrates wiltr?third-party metrics platforms to automatically scale up and down the number of instances of the Spring Boot application.

Spring Boot Actuator (https://www.baeldung.com/spring-boot-actuators)

Q32. What is the purpose of the @ContextConfiguration annotation in a JUnit Test?

- [ ] It introspects the local machine and automatically provisions resources based on certain contextual configuration files.
- [ ] It automatically generates comments for annotated classes on autowired dependencies to provide additional context about dependencies.
- [✓] It defines metadata at the class-level to determine how to load or configure an ApplicationContext in Spring integration tests.
- [ ] It automatically generates JavaDocs for annotated classes to provide additional context about the purpose of the class.

@ContextConfiguration Example in Spring Test (https://www.concretepage.com/spring-5/contextconfiguration-example-spring-test)

Q33. How are authentication and authorization different?

- [ ] Authentication is the act of granting access to specific resources and functions based on config settings. Authorization is the act of introspecting a user's credentials to ensure they are not impersonating another user.
- [ ] Authentication is the act of verifying certain resources and functions are actually valid. Authorization is the act of verifying a user's credentials have not expired.
- [ ] Authentication is the act of verifying that certain resources and functions actually exist in the database. Authorization is the act of verifying a user's credentials to ensure they are valid.
- [✓] Authentication is validating that users are who they claim to be. Authorization is granting access to specific resources and functions.

Q34. What is the purpose of the @RequestBody annotation?

- [ ] to create a ThreadLocal byte stream that allows a request to be encoded for reading directly into a database

- ☐ to automatically generate a ThreadLocal byte stream from the body of a request that allows a request to scanned for security risks
- ✓ to indicate whether an annotated handler method parameter should be bound to the web request body, which is converted by an HttpMessageConverter
- ☐ to automatically validate the characters contained in a request to ensure that they are a valid character encoding

Q35. What is the DispatcherServlet and what is its function?

- ☐ The DispatcherServlet process daemon assigns a separate Web Servlet Container process to each HTTP request that comes into the web server.
- ✓ It is a servlet that dispatches HTTP requests to registered handlers/controllers for processing.
- ☐ The DispatcherServlet API assigns a separate Web Servlet Node process to each additional HTTP request that comes into the web server.
- ☐ It is a servlet that dispatches an array of background daemon processes that allocate memory and CPU cycles to each request.

Q36. What is Spring Boot autoconfiguration?

- ☐ It triggers a local automated review of configuration files such as web.xml and detects possible security issues or automatically resolves circular dependencies.
- ☐ It triggers an automated review of configuration by a web-based agent that reviews your existing web.xml file and detects possible security issues.
- ✓ It's an opinionated, intelligent method of introspecting an app to configure beans that are likely to be needed. This configuration can be overridden over time with manual configuration.
- ☐ It provides plug-in functionality while editing your web.xml and other config files that will autocomplete common dependencies while typing.

Q37. Which are valid steps to take to enable JPA in Spring Boot?

- ✓ Add an @EnableJpaRepositories annotation to your configuration class and create a Spring Data Repository.
- ☐ Add an @EnableJpaEntities annotation to your configuration class, create a Spring Data YAML configuration file, and manually update the Spring XML config files to define your repository locations.
- ☐ Add an @EnableDbFunctionality annotation to your configuration class, create a Spring Data XML configuration file, and manually update the Spring factories file to define your repositories.
- ☐ Add an @InitJpaEntities annotation to your configuration class, create a Spring Data properties configuration file, and manually update the Spring startup parameters to define your repository locations.

Q38. What is a transaction in the context of Spring Data?

- ☐ a version-controlled schema change that is applied to a database
- ✓ a sequence of actions representing a single unit of work managed as a single operation that can be either committed or rolled back
- ☐ an encoded message and response between various shards of a database
- ☐ an exchange or interaction between various worker nodes in a multithreaded environment

Q39. Modularization of a concern that cuts across multiple classes is known as a(n)____.

- ☐ multiclass
- ✓ aspect
- ☐ crosscut
- ☐ sidecut

Q40. How do you inject a dependency into a Spring bean?

- ☐ Use field injection.
- ☐ Annotate a Setter method with the @Autowired annotation.
- ☐ Specify parameters in the constructor with an optional @Autowired annotation.
- ✓ Any of the above.

Q41. Consider the properties file application.properties. How would you load the property my.property?

```
my.property=Test
```

- ☐ A

```
@Prop("${my.property}")
private String val;
```

- ☐ B

```
@GetVal("my.property")
private String val;
```

- ☐ C

```
@GetProperty("${my.property}")
private String val;
```

- ✓ D

```
@Value("${my.property}")
private String val;
```

Q42. What is a bean in the context of Spring?

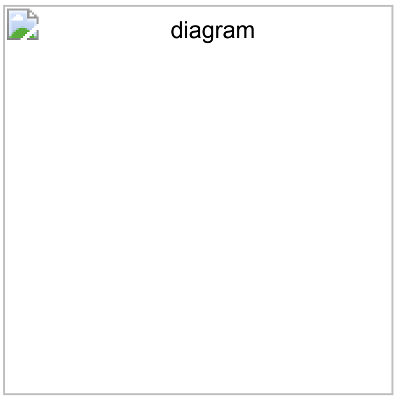
- ✓ a managed dependency defined by configuration metadata that can be injected into downstream classes
- ☐ a binary-encoded, agnostic, named entity that is translatable between different data formats

- a payload that is transferable between different services in a Service-Oriented Architecture (SOA)
- a discrete piece of data that is encoded in a binary format for persisting to a file system

Q43. Which property is given precedence by Spring?

- application properties located in an application.properties file outside the application.jar
- @PropertySource annotations on configuration classes
- [✓ profile-specific application-.properties files]
- application properties located in an application.properties file inside the application.jar

Q44. In the Spring Bean lifecycle pictured, what should the third step of the process be?



- Persist bean definitions into a database
- [✓ Instance bean objects]
- De-normalize bean definition
- Use reflection to inject bean objects into the servlet container

Q45. What Spring Boot property is used to set the logging level for the entire application in the application.properties file?

- logging.settings
- log.level
- root.logger.level
- [✓ logging.level.root]

[Logging in Spring Boot (https://www.baeldung.com/spring-boot-logging)](https://www.baeldung.com/spring-boot-logging)

Q46. What is a Spring bean uniquely identified?

- an autogenerated UUID
- [✓ a unique String name]
- an auto-incremented Integer ID
- an ID derived from its location in memory

Q47. What is the difference between a JAR and a WAR distribution in Spring Boot?

- Spring Boot can create a self-executable WAR file that runs without needing a servlet container. A JAR file has to be deployed to an existing web container such as Tomcat with separate files for dependencies.

- ☐ Spring Boot can create a JAR file that contains bytecode that interacts directly with the OS without needing a servlet container. A WAR file has to be deployed to an existing web container such as Tomcat with separate files for dependencies.
- ☐ The Spring Boot JAR file will be deployed to a Servlet container such as Tomcat on an existing running web server locally. The WAR file will be deployed to a cloud-based servlet container.
- ✓ Spring Boot can create a self-executable JAR file that contains all the dependencies and an embedded Servlet container. A WAR file has to be deployed to an existing web container such as Tomcat.

Q48. How does the transaction propagation setting impact the behavior of transactions?

- ☐ It ensures that transactions that are commited to the database are propagated to all the shards of a clustered database system.
- ✓ It enforces that a logical transaction scope is created for each method that determines rollback-only status, with an outer transaction scope being logically independent from the inner transaction scope.
- ☐ It guarantees that transactions that are submitted to the database cluster are propagated to all the nodes of a clustered database cloud.
- ☐ None of the above

Q49. What is printed when this code is run as a @SpringBootApplication?

```
@Component
public class Test implements InitializingBean {
     @Autowired
     ApplicationContext context;

     private TestService service;
     public void setService(TestService service) {
          this.service = service;
     }

     @Override
     public void afterPropertiesSet() throws Exception {
          System.out.print(context.containsBean("testService") + " ");
          System.out.println(service.getClass());
     }
}
@Service
class TestService {}
```

- ✓ a null pointer stacktrace
- ☐ true null
- ☐ true package.TestService
- ☐ false package.TestService

Explanation: missing `@Autowired` on `private TestService service` or on the setter

Q50. To register a custom filter that applies only to certain URL patterns, you should remove the _ annotation from the filter class and register a @Bean of type _ in Spring @Configuration.

- [ ] @RequestMapping; WebRequest
- [ ] @Controller; URLFilter
- [ ] @WebFilter; ServletContextInitializer
- [✓] @Component; FilterRegistrationBean

Q51. What is the correct term for each definition bellow?

1. A predicate that matches join points.
2. A point during the execution of a program, such as the execution of a method or the handling of an exception.
3. An action taken by an aspect at a particular join point.

- [ ]

```
1. Pointcut
2. Advice
3. Join point
```

- [ ]

```
1. Join point
2. Pointcut
3. Advice
```

- [ ]

```
1. Advice
2. Pointcut
3. Join point
```

- [✓]

```
1. Pointcut
2. Join point
3. Advice
```

Q52. How should passwords be stored?

- [✓] Passwords should be hashed using an adaptive one-way function such as bcrypt.
- [ ] Passwords can be stored in a BASE64 encoded format if they are stored in a private database.
- [ ] Passwords should be salted and hashed using the MD5 algorithm.

- ☐ Passwords should be hashed using the SHA-1 algorithm, then salted to provide defence against rainbow table attacks.

  Explanation: sha-1 is not considered secure anymore: https://en.wikipedia.org/wiki/SHA-1#Attacks (https://en.wikipedia.org/wiki/SHA-1#Attacks) . With bcrypt you can select more complex hashes https://en.wikipedia.org/wiki/Bcrypt (https://en.wikipedia.org/wiki/Bcrypt)

Q53. What methods does this Pointcut expression reference?

```
@target(com.linkedin.annotation.Loggable)
```

- ✓ any join point where the target object has a @Loggable annotation
- ☐ any join point where the executing method has a @Loggable annotation
- ☐ any method that implements Loggable
- ☐ any method that extends Loggable

Difference between @target and @within (Spring AOP) (https://stackoverflow.com/questions/51124771/difference-between-target-and-within-spring-aop)

Q54. What is printed when this code is run as a @SpringBootApplication?

```
@Component
public class Test implements InitializingBean {
     @Autowired
     ApplicationContext context;

     @Autowired
     SimpleDateFormat formatter;

     @Override
     public void afterPropertiesSet() throws Exception {
          System.out.println(context.containsBean("formatter"));
          System.out.println(formatter.getClass());
     }
}
@Configuration
class TestConfig2 {
    @Bean
    public final SimpleDateFormat formatter() {
        return new SimpleDateFormat();
    }
}
```

- ☐ true
  class java.text.SimpleDateFormat

- ☐ true
  SimpleDateFormat
- ☐ a NullPointerException stacktrace
- ☑ a BeanDefinitionParsingException stacktrace

Explanation: `@Bean`-method in `@Configuration` must be overridable. Remove the `final` keyword to fix.

Q55. What is the purpose of a web application context?

- ☐ Configures a web application that is able to be deleted and re-created during runtime through hot swap. It adds a **recreateContext()** method and defines a root WebDaemon that must be bound to in the bootstrap process.
- ☐ It configures a Spring application that is able to be modified on the fly during runtime through bytecode re-encoding. Also it adds an **updateContext()** method and defines a root WebServlet that must be bound to in the bootstrap process.
- ☑ It provides configuration for a web application that is read-only while running. Also, it adds a **getServletContext()** method and defines an attribute name to which the root context must be bound in the bootstrap process.
- ☐ It provides configuration for a Spring application that is updatable on the fly during runtime through bytecode weaving. Also it adds an **updateServletContext()** method and defines a root servlet that must be bound to in the bootstrap process.

Q56. What is Spring AOP?

- ☐ Aspect-Oriented Programming allows you to define different cross-cutting aspects that isolate beans to be available only in certain environments and profiles.
- ☐ Aspect-Oriented Programming enables you to perform profiling, which lets you develop different cross-cutting JVM performance-tuning profiles for different aspects of your applications.
- ☑ Aspect-Oriented Programming enables the modularization of cross-cutting concerns so that repeated boilerplate logic, such as logging code, does not pollute business logic.
- ☐ Aspect-Oriented Programming enables you to persist cross-cutting data across modularized shards of your database.

Q57. Assuming username and password authentication is in place, what method on the Authentication object can be used to obtain the username?

- ☑ getPrincipal
- ☐ getUsername
- ☐ getUser
- ☐ getDn

Q58. Assuming no additional configuration is provided, what is the first selection criteria Spring uses to choose a bean when autowiring a property?

- ☑ bean name
- ☐ bean type
- ☐ bean size
- ☐ None of the above

Q59. What is the result of calling the map controller method using the following HTTP request?

```
POST localhost:8080/map
{"b" : "b", "d" : "d"}
```

```
@RestController
public class SampleController {
    @RequestMapping("/map")
    public String map(@RequestBody SampleObject sampleObject) {
        return sampleObject.getB() + sampleObject.getC();
    }
}
```

```
public class SampleObject {

    String b;
    String c;

    public String getB() { return b; }

    public void setB() { this.b = b; }

    public String getC() { return c; }

    public void setC() { this.c = c; }
}
```

- ☐ An InvalidRequestBodyException is thrown at runtime.
- ☐ A MissingPropertyException is thrown at runtime.
- [✓] The text "bnull" is returned in the response body.
- ☐ The text "a" is returned in th response body.

Q60. What effect does private static have on the object service below?

```
@SpringBootApplication
public class Question14 {
    @Autowired
    private static Service service;

    public static void main(String[] args) {
        SpringApplication.run(Question14.class, args);
    }
}

@Component
class Service {}
```

- ☐ The application will result in a compile error because you can't autowire a private variable.
- ☐ The application will compile and run, and service will have its dependency correctly injected by Spring.
- ☑ The application will compile and run, but service will not be autowired because you cannot autowire a static class member.
- ☐ The application will result in a compile error because you attempted to autowire a static variable.

Q61. What is a security context?

- ☑ The security context includes details of the principal currently using the app, which is stored by default in a `ThreadLocal` in an `Authentication` object.
- ☐ The security context holds a list of all users and their encrypted passwords in memory and a list of resources that users are able to access.
- ☐ The security context includes information about safe network IDs and IP addresses that are able to access the system.
- ☐ The security context includes information about permissions on the local file system describing how local file resources can be accessed.

Q62. How might you map an incoming request to a controller method?

- ☐ Annotate a Controller class with `@Controller`. Then, using a specific naming convention for the methods, the `RequestMappingHandlerAdapter` will automatically configure your endpoints with the proper HTTP verb and URI.
- ☐ Register a controller as a bean. Then, using a specific naming convention for the methods, the `RequestMappingHandlerAdapter` will automatically configure your endpoints based on values from the YAML config file.
- ☑ Annotate a controller method with `@RequestMapping`, or a HTTP verb-specific annotation with a String URI pattern parameter (and other params as needed), which is supported through a `RequestMappingHandlerMapping/Adapter`.
- ☐ Register a controller as a bean. Then, using a specific naming convention for the methods, the RequestMappingHandlerAdapter will automatically configure your endpoints based on values passed into the bean definition.

Spring RequestMapping (https://www.baeldung.com/spring-requestmapping). Spring does not use naming conventions for web requests (unlike e.g. for the Data Repositories)

Q63. What methods does the Pointcut expression below reference?

```
execution(* setter*(..))
```

- ☐ any method with a name that contains the String "setter" with a single parameter
- ☐ any method with a name that begins with String "setter" with a single parameter
- ✓ any method with a name that begins with String "setter"
- ☐ any method with a name that contains the String "setter"

baeldung (https://www.baeldung.com/spring-aop-pointcut-tutorial#1-execution)

Q64. What pattern does Spring MVC implement to delegate request processing to controllers?

- ✓ Front Controller
- ☐ Facade
- ☐ Reactive Chain
- ☐ Observer

Q65. What methods does this Pointcut expression?

```
within(com.linkedin.service..*)
```

- ☐ any join point only within the service package
- ✓ any join point within the service package or one of its subpackages
- ☐ any method in a class that autowires a service bean
- ☐ This is not valid Pointcut expression

Q66. What is the output from invoking this Actuator endpoint in an unmodified Spring Boot application generated using Spring Intializr?

```
/shutdown
```

- ✓ The request would fail because the endpoint is disabled by default.
- ☐ The Spring Boot app would return a URL and a list of currently running processes, and ask you to confirm the shutdown command.
- ☐ The Spring Boot app would return a URL to confirm the shutdown command.
- ☐ The Spring Boot app would begin shutting down.

Reason: By default, all the endpoints are enabled in Spring Boot Application except /shutdown; this is, naturally, part of the Actuator endpoints.

Q67. How can you access the application context in a Spring integration test?

- ☐ The context is present in a ThreadLocal so you can just call getSpringApplicationContextInstance() anywhere in a test to get the current context.

- ✓ Test classes can implement the ApplicationContextAware interface or autowire the ApplicationContext.
- ☐ The context is automatically injected in every test so you can just call getSpringApplicationContextInstance() anywhere in a test to get the current context.
- ☐ You can just add the parameter Context context to any method so that the context is automatically wired in.

[spring(dot)io (https://spring.io/blog/2011/06/21/spring-3-1-m2-testing-with-configuration-classes-and-profiles#:%7E:text=By%20default%20the%20ApplicationContext%20is,%40Resource%20%2C%20or%20%40Inject%20.)](https://spring.io/blog/2011/06/21/spring-3-1-m2-testing-with-configuration-classes-and-profiles#:%7E:text=By%20default%20the%20ApplicationContext%20is,%40Resource%20%2C%20or%20%40Inject%20.)

Q68. What interface can be specified as a parameter in a controller method signature to handle file uploads?

- ☐ FilePath
- ✓ MultipartFile
- ☐ File
- ☐ MvcFile

Q69. What is the purpose of this endpoint?

```
@GetMapping("api/v1/domain/resource/{id}")
public Pojo getPojo(@PathVariable("id") String id) {
  return testService.getPojo(id);
}
```

- ☐ This endpoint is designed to delete an object of the typetype Pojo with the passed in path variable
- ✓ This endpoint is designed to return JSON mapped to the object type Pojo with the passed in id.
- ☐ This endpoint returns the path variable that describes the Pojo's location within the container.
- ☐ This endpoint is designed to update an existing resource of the object type Pojo with the passed in id.

Q70. What property can be used to set the active Spring profiles

- ☐ active.profile
- ☐ active.spring.profiles
- ✓ spring.profiles.active
- ☐ profiles

Q71. Which statement is true regarding loading and instantiation of Spring factories?

- ☐ During startup, the SpringFactoryInitializr collects all files in the CONFIG-INF directory from each dependency and downloads binaries to run each file.
- ✓ During startup, the SpringFactoriesLoader gets a list of config and collects all the files in META-INF directory from dependencies. Then it builds a composite list for application context configurations.
- ☐ During shutdown, the SpringFactoryDestructor collects all the files in META-INF directory from each dependency and begins shutting down each thread and process.
- ☐ During startup and shutdown, the SpringFactoryInitializr downloads project configs for all configured dependencies.

Q72. What methods does this Pointcut expression reference?

```
execution(* com.linkedin.TestService.*(..))
```

- [✓] all methods of classes in the com.linkedin.TestService package
- [ ] all methods of classes in the com.linkedin.TestService package annotated whith @Service
- [ ] This Pointcut is not valid.
- [ ] all methods defined by the TestService interface

Q73. When configuring an application, which configuration is given precedence by Spring?

- [ ] profile specific application-.properties files
- [ ] Java System Properties
- [ ] application properties located in an application.properties file inside the application.jar
- [✓] profile specific application-.properties files located outside the application.jar

Q74. What interface is used to represent a permission in Spring Security?

- [ ] GrantedAuthority
- [ ] SecurityChain
- [ ] PermissionMatrix
- [✓] AccessRule

Q75. What is the difference between constructor injection and setter injection?

- [ ] Constructor injection overrides setter injection.
- [ ] Setter injection creates a new instance if any modification occurs.
- [✓] You can't use constructor injection for partial injection.
- [ ] Constructor injection is more flexible than setter injection.

[Explanation] There are many key differences between constructor injection and setter injection.

Partial dependency: can be injected using setter injection but it is not possible by constructor. Suppose there are 3 properties in a class, having 3 arg constructor and setters methods. In such case, if you want to pass information for only one property, it is possible by setter method only. Overriding: Setter injection overrides the constructor injection. If we use both constructor and setter injection, IOC container will use the setter injection. Changes: We can easily change the value by setter injection. It doesn't create a new bean instance always like constructor. So setter injection is flexible than constructor injection.

Q76. Which println would you remove to stop this code from throwing a null pointer exception?

```
@Component
public class Test implements InitializingBean {
     @Autowired
     ApplicationContext context;
     @Autowired
     static SimpleDateFormat formatter;

     @Override
     public void afterPropertiesSet() throws Exception {
          System.out.println(context.containsBean("formatter") + " ");
          System.out.println(context.getBean("formatter").getClass());
          System.out.println(formatter.getClass());
          System.out.println(context.getClass());
     }
}
@Configuration
class TestConfig {
     @Bean
     public SimpleDateFormat formatter() {
          return new SimpleDateFormat();
     }
}
```

- ☐ formatter.getClass()
- ☐ context.containsBean("formatter")
- ✓ context.getBean("formatter").getClass()
- ☐ context.getClass()

Explanation: Here only one line can throw NPE. Calling getClass() from context.getBean("formatter") can potentially throw NPE if context.getBean("formatter") will return null.

Q77. What is the default rollback policy?

- ☐ A rollback is triggered during any error that occurs during the transaction.
- ☐ When an instance or subclass of Exception is thrown, this triggers a rollback, while any runtime exceptions do not.
- ☐ Anytime an instance or subclass of Throwable is thrown, this triggers a rollback.
- ✓ When an instance or subclass of RuntimeException is thrown, this triggers a rollback, while any checked Exception does not.

Q78. What is the difference between a CrudRepository and a JpaRepository?

- ☐ The CrudRepository extends the PagingAndSortingRepository, while the JpaRepository does not.
- ✓ The CrudRepository exposes a superset of interface methods containing every datastore-specific method supported by Spring data. The JpaRepository contains only those specific to Spring Data JPA.

- ☐ The CrudRepository is a base interface within Spring Data core that exposes a dedicated set of functions. The JpaRepository is a store-specific interface that exposes functionality specific to JPA.
- ☐ The CrudRepository is part of the Java EE API, while JpaRepository is specific to Spring Data.

Q79. What is the security filter chain?

- ☐ It's a security filter chain that provides authentication with manual intervention such that multiple administrators of the system are able to approve users with auditability and traceability.
- ☐ It's a series of user-completed activities—such as password authorization, token verification, and many others—that require multiple factors of authentication to increase the level of security in the system.
- ☑ It's a servlet filter chain where each filter has a specific responsibility such as security context, user and password authorization, exception translation, or filter security interception, processed in order.
- ☐ It's a security filter chain that consumes multiple factors of authentication—such as password, token verification, biometrics, and IP whitelisting—to successfully log a user into the system.

Q80. Which is not a valid stereotype annotation?

- ☐ @Component
- ☐ @Service
- ☑ @HtmlController
- ☐ @Controller

Q81. Which statement is true regarding loading and instantiation of Spring factories?

- ☑ During startup, the SpringFactoriesLoader gets a list of configs and collects all the files in META-INF directory from dependencies. Then it builds a composite list for application context configurations.
- ☐ During shutdown, the SpringFactoryDestructor collects all the files in META-INF directory from each dependency and begins shutting down each thread and process.
- ☐ During startup, the SpringFactoryInitializr collects all files in the CONFIG-INF directory from each dependency and downloads binaries to run each file.
- ☐ During startup and shutdown, the SpringFactoryInitializr downloads project configs for all configured dependencies.

Q83. What is a transaction isolation level?

- ☐ executing each transaction in its own dedicated threadpool to facilitate thread isolation
- ☐ facilitating each transaction to occur on its own dedicated vCPU to guarantee throughput
- ☑ the level of visibility and access a transaction has to the units of work of other transactions such as uncommitted writes
- ☐ executing each transaction on its own process to provide resource isolation

Q84. What does the statement "Spring offers fully-typed advice" mean?

- ☐ You declare the parameters you need in the advice signature rather than work with Object[] arrays.
- ☐ You work with a collection of Objects that need to be explicitly casted.
- ☑ You work with an array of a Generic type T[] instead of Object[] arrays.
- ☐ You are able to undo type erasure in the Object[] that is exposed

Q84. Which are considered to be typical, common, cross-cutting concerns that would be a good fit for AOP? (Choose 3)

```
- A. Creating SQL queries
- B. Logging
- C. Filtering, sorting and transforming data
- D. Transaction management
- E. Audit logging
- F. Business logic
```

- ☐ A, D, F
- ☐ D, E, F
- ☐ A, B, F
- [✓ B, D, E]

Q85. Which of the Service implementations will be created first?

```
@SpringBootApplication
public class App {

     @Autowired
     Service service;

     public static void main(String[] args) {
          SpringApplication.run(App.class, args);
     }
}
@Primary
@Component
class Service2Impl implements Service {

    Service2Impl() {
        System.out.println("Service2");
    }
}

@Component("Service")
class Service1Impl implements Service {

    Service1Impl() {
        System.out.println("Service1");
    }
}

interface Service{}
```

- ☐ Service1
- ✓ Service2
- ☐ A NullPointerException is thrown at runtime.
- ☐ There is no way to know until runtime.

Explaination: Primary indicates that a bean should be given preference when multiple candidates are qualified to autowire a single-valued dependency (https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Primary.html)

Q86. What methods does this Pointcut expression reference?

```
     execution(* com.linkedin.service..*.*(..))
```

- ✓ all methods defined in the service package or one of it's subpackages
- ☐ all methods defined in the service package

- ☐ all methods defined in a service interface
- ☐ The pointcut is invalid.

Q87. Which is not a core facet of Spring's ecosystem?

- ☐ Spring Data
- ☐ Spring MVC
- ✓ Spring Bootstrap
- ☐ Spring Cloud

Q88. A_ is a key-value map of data used to render the page, and the _ is a template of the page that is filled with data.

- ☐ model; view
- ☐ hashmap; serviet
- ✓ view; model
- ☐ request; view