Core Spring Framework 4.x and certification udemy course

Content and related file locations in sample applications

CONTAINER, DEPENDENCY, AND IOC

1. What is dependency injection and what are the advantages?
   application-context.xml, spring.container.dependency.ioc.javaconfig.JavaConfig
2. What is an interface and what are the advantages of making use of them in Java?
   spring.bean.Log
3. What is meant by “container” and how do you create one?
   spring.container.dependency.ioc.XMLconfig.MainApp
4. What is the concept of a container and what is its lifecycle?
5. Dependency injection using Java configuration
   spring.container.dependency.ioc.javaconfig.JavaConfig
6. Dependency injection in XML, using constructor or setter injection
   application-context.xml
7. Dependency injection using annotations (@Component, @Autowired)
   spring.bean.DatabaseService
8. Component scanning, Stereotypes and Meta-Annotations
   spring.container.dependency.ioc.javaconfig.JavaConfig
9. Scopes for Spring beans. What is the default?
   application-context.xml
10. What is an initialization method and how is it declared in a Spring bean?
    spring.bean.DatabaseService, application-context.xml
11. What is a destroy method, how is it declared and when is it called?
    spring.bean.DatabaseService, application-context.xml
12. What is a BeanFactoryPostProcessor and what is it used for?
13. What is a BeanPostProcessor and how is the difference to a BeanFactoryPostProcessor? What do they do? When are they called?
14. Are beans lazily or eagerly instantiated by default? How do you alter this behavior?
    spring.container.dependency.ioc.javaconfig.JavaConfig, application-context.xml
15. What does component-scanning do?
    spring.container.dependency.ioc.javaconfig.JavaConfig, application-context.xml
16. What is the behavior of the annotation @Autowired with regards to field injection, constructor injection and method injection?
    spring.bean.DatabaseService, spring.bean.LoginService, spring.bean.RegisterService
17. How does the @Qualifier annotation complement the use of @Autowired?
    spring.bean.DBLogService, spring.bean.RegisterService
18. What is the role of the @PostConstruct and @PreDestroy annotations? When will they get called?
    spring.bean.DatabaseService
19. What is a proxy object and what are the two different types of proxies Spring can create?
20. What is the power of a proxy object and where are the disadvantages?
21. What are the limitations of these proxies (per type)?
22. How do you inject scalar/literal values into Spring beans?
    spring.bean.MailService, application-context.xml
23. How are you going to create a new instance of an ApplicationContext?
    spring.container.dependency.ioc.javaconfig.MainApp,
    spring.container.dependency.ioc.XMLconfig.MainApp
24. What is a prefix?
    application-context.xml, spring.container.dependency.ioc.javaconfig.JavaConfig
25. What is the lifecycle on an ApplicationContext?
26. What does the @Bean annotation do?
    spring.container.dependency.ioc.javaconfig.JavaConfig
27. How are you going to create an ApplicationContext in an integration test or a JUnit test?
    spring.test.TestMailService, spring.test.TestMailServiceAbstractJUnit4SpringContextTests,
    spring.test.TestMailServiceApplicationContextAware

28. What do you have to do, if you would like to inject something into a private field?
spring.bean.RegisterService
29. What are the advantages of JavaConfig? What are the limitations?
30. What is the default bean id if you only use "@Bean"?
application-context.xml
31. Can you use @Bean together with @Profile?
spring.container.dependency.ioc.javaconfig.JavaConfig
32. What is Spring Expression Language (SpEL for short)?
application-context.xml, spring.bean.MailService
33. What is the environment abstraction in Spring?
34. How do you configure a profile. What are possible use cases where they might be useful?
spring.container.dependency.ioc.javaconfig.JavaConfig, application-context-profile.xml
35. How many profiles can you have?
application-context-profile.xml
36. What is an inner bean? Why does it not have a bean id? Can it be reused?
application-context.xml
37. What is bean definition inheritance?
application-context.xml
38. What happens if the parent bean is not abstract?
39. What is a ProperyPlaceholderConfigurer used for?
spring.container.dependency.ioc.javaconfig.JavaConfig, application-context-profile.xml
40. When do you have to implement the FactoryBean interface?
spring.util.ComplexBeanFactoryBeanImplementation
41. How do you enable JSR-250 annotations like @PostConstruct?
spring.container.dependency.ioc.javaconfig.JavaConfig, application-context-profile.xml
42. Why are you not allowed to annotate a final class with @Configuration
43. Which application context implementation are you using when you have used @Configuration?
44. Why must you have to have a default constructor in your @Configuration annotated class?
45. Why are you not allowed to annotate final methods with @Bean?
46. What is the preferred way to close an application context?
spring.container.dependency.ioc.javaconfig.JavaConfig,
spring.container.dependency.ioc.XMLconfig.MainApp
47. How can you create a shared application context in a JUnit test?
spring.test.TestJavaConfigProfile

ASPECT ORIENTED PROGRAMMING

48. What is the concept of AOP? Which problem does it solve?
49. What is a pointcut, a join point, an advice, an aspect, weaving?
50. How does Spring solve (implement) a cross cutting concern?
spring.aspect.AspectManager, aspect-config.xml
51. Which are the limitations of the two proxy-types?
52. How many advice types does Spring support. What are they used for?
spring.aspect.AspectManager, aspect-config.xml
53. What do you have to do to enable the detection of the @Aspect annotation?
54. Name three typical cross cutting concerns.
55. What two problems arise if you don't solve a cross cutting concern via AOP?
56. What does @EnableAspectJAutoProxy do?
spring.container.dependency.ioc.javaconfig.JavaConfig
57. What is a named pointcut?
spring.aspect.AspectManager, aspect-config.xml
58. How do you externalize pointcuts? What is the advantage of doing this?
spring.aspect.AspectManager,spring.aspect.ExternalizeAspectExpression
59. What is the JoinPoint argument used for?
spring.aspect.AspectManager
60. What is a ProceedingJoinPoint?
spring.aspect.AspectManager

61. How are the five advice types called?
spring.aspect.AspectManager
62. Which advice do you have to use if you would like to try and catch exceptions?
spring.aspect.AspectManager

JDBC, TRANSACTIONS, AND ORM

63. What is the difference between checked and unchecked exceptions?
64. Why do we (in Spring) prefer unchecked exceptions?
65. What is the data access exception hierarchy?
66. How do you configure a DataSource in Spring? Which bean is very useful for development?
application-context-jdbc.xml
67. What is the Template design pattern and what is the JDBC template?
spring.jdbc.transactions.orm.UserManager
68. What is a callback? What are the three JdbcTemplate callback interfaces described in the notes? What are they used for? (You would not have to remember the interface names in the exam, but you should know what they do if you see them in a code sample).
spring.jdbc.transactions.orm.UserManager
69. Can you execute a plain SQL statement with the JDBC template?
spring.jdbc.transactions.orm.UserManager
70. Does the JDBC template acquire (and release) a connection for every method called or once per template?
71. Is the JDBC template able to participate in an existing transaction?
72. What is a transaction? What is the difference between a local and a global transaction?
73. Is a transaction a cross cutting concern? How is it implemented in Spring?
74. How are you going to set up a transaction in Spring?
spring.jdbc.transactions.orm.UserManager.DBLogService
75. What does @Transactional do? What is the PlatformTransactionManager?
application-context-jdbc.xml
76. What is the TransactionTemplate? Why would you use it?
spring.jdbc.transactions.orm.UserManagerTransactionTemplate
77. What is a transaction isolation level? How many do we have and how are they ordered?
spring.jdbc.transactions.orm.UserManager
78. How does the JdbcTemplate support generic queries? How does it return objects and lists/maps of objects?
spring.test.TestJDBC
79. What does transaction propagation mean?
spring.jdbc.transactions.orm.UserManager
80. What happens if one @Transactional annotated method is calling another @Transactional annotated method on the same object instance?
81. Where can the @Transactional annotation be used? What is a typical usage if you put it at class level?
spring.jdbc.transactions.orm.UserManager
82. What does declarative transaction management mean?
application-context-jdbc.xml
83. What is the default rollback policy? How can you override it?
84. What is the default rollback policy in a JUnit test, when you use the SpringJUnit4ClassRunner and annotate your @Test annotated method with @Transactional?
spring.test.TestJDBC
85. Why is the term "unit of work" so important and why does JDBC AutoCommit violate this pattern? What does JPA mean - what is ORM? What is the idea behind an ORM?
86. What is a PersistenceContext and what is an EntityManager. What is the relationship between both? Why do you need the @Entity annotation. Where can it be placed?
87. What do you need to do in Spring if you would like to work with JPA?
application-context-persistence.xml,spring.test.TestJPAPersistence
88. Are you able to participate in a given transaction in Spring while working with JPA?
spring.jdbc.transactions.orm.UserManager
89. What does @PersistenceContext do?
90. What are disadvantages or ORM? What are the benefits?

91. What is an "instant repository"? (hint: recall Spring Data)

SPRING MVC

92. MVC is an abbreviation for a design pattern. What does it stand for and what is the idea behind it?
93. Do you need spring-mvc.jar in your classpath or is it part of spring-core?
94. What is the DispatcherServlet and what is it used for?
web.xml
95. Is the DispatcherServlet instantiated via an application context?
96. What is the root application context? How is it loaded?
97. What is the @Controller annotation used for? How can you create a controller without an annotation?
98. What is the ContextLoaderListener and what does it do?
web.xml
99. What are you going to do in the web.xml. Where do you place it?
web.xml
100. How is an incoming request mapped to a controller and mapped to a method?
spring.controller.RegisterController, spring.controller.XMLController
101. What is the @RequestParam used for?
spring.controller.UserController, admin.jsp, user.jsp
102. What are the differences between @RequestParam and @PathVariable?
spring.controller.UserController, admin.jsp, user.jsp
103. What are some of the valid return types of a controller method?
104. What is a View and what's the idea behind supporting different types of View?
105. How is the right View chosen when it comes to the rendering phase?
106. What is the Model?
spring.controller.UserController
107. Why do you have access to the model in your View? Where does it come from?
spring.controller.UserController, admin.jsp, user.jsp
108. What is the purpose of the session scope?
spring.controller.UserController, admin.jsp
109. What is the default scope in the web context?
110. Why are controllers testable artifacts?
111. What does the InternalResourceViewResolver do?
mvc-config.xml

SECURITY

112. What is the delegating filter proxy?
web.xml
113. What is the security filter chain?
114. In the notes several predefined filters were shown. Do you recall what they did and what order they occurred in?
115. Are you able to add and/or replace individual filters?
116. Is it enough to hide sections of my output (e.g. JSP-Page)?
117. Why do you need the intercept-url?
security.xml
118. Why do you need method security? What type of object is typically secured at the method level (think of its purpose not its Java type).
119. Is security a cross cutting concern? How is it implemented internally?
120. What do @Secured and @RolesAllowed do? What is the difference between them?
mvc-config.xml, spring.controller.UserController
121. What is a security context?
122. In which order do you have to write multiple intercept-url's?
security.xml
123. How is a Principal defined?
124. What is authentication and authorization? Which must come first?
125. In which security annotation are you allowed to use SpEL?
mvc-config.xml, spring.controller.UserController

126. Does Spring Security support password hashing? What is salting?
security.xml

MESSAGING

127. What is JMS?
128. What are the advantages of messaging?
129. What is a JmsTemplate used for? How are you going to create it?
jms-config.xml, spring.jms.MessageReceiver, spring.jms.MessageSender
130. Are you able to work asynchronous in JMS/Spring-Messaging?
131. What is a broker and why does it decouple your application components?
132. What is the difference between a queue and a topic? Name some real world examples?
133. Are the messages stored in the broker? How long?
134. What is the JmsTemplate and which callback does it have?
spring.controller.MessageController, spring.jms.MessageSender, spring.jms.JMSListener2
135. What are the defaults if you do not provide a callback implementation to the JmsTemplate?
136. What is a JMS listener? Which method does it force you to implement? When will the method get called?
jms-config.xml, spring.controller.MessageController, spring.jms.JMSListener2
137. What is a message, what is a Destination and what is a MessageProducer?
138. What is a JMS session? What do you have to do if you would like to work in or participate in a transaction?
139. How can you work asynchronously in Spring JMS when you cannot implement the MessageListener interface? What is a message driven POJO?
jms-config.xml, spring.jms.JMSListener1
140. If you implement the MessageListener interface - who is calling your onMessage method? What do you have to configure to make this working?
jms-config.xml, spring.jms.JMSListener2
141. What do you have to do if you would like to work transactionally?

REST

142. What does REST stand for?
143. What is a resource?
144. What are safe REST operations?
145. What are idempotent operations? Why is idempotency important?
146. Is REST scalable and/or interoperable?
147. What are the advantages of the RestTemplate?
148. Which HTTP methods does REST use?
spring.controller.AdminRestController
149. What is an HttpMessageConverter?
spring.controller.AdminRestController, spring.bean.DBLog
150. Is REST normally stateless?
151. What does @RequestMapping do?
spring.controller.AdminRestController
152. Is @Controller a stereotype? Is @RestController a stereotype?
153. What is the difference between @Controller and @RestController?
spring.controller.RestControllerExample
154. When do you need @ResponseBody?
spring.controller.AdminRestController
155. What does @PathVariable do?
spring.controller.AdminRestController
156. What is the HTTP status return code for a successful DELETE statement?
157. What does CRUD mean?
158. Is REST secure? What can you do to secure it?
159. Where do you need @EnableWebMVC?
160. Name some common http response codes. When do you need @ResponseStatus?
spring.controller.RestControllerExample
161. Does REST work with transport layer security (TLS)?
162. Do you need Spring MVC in your classpath?