Professional VMware Spring

Exam Details (Last Updated: 02/16/2021)

Spring Professional certification is a 50-multiple-choice exam, with a passing score of 76% correctness. (In other words, you must answer 38 or more questions correctly.) Candidates are given 90 minutes to complete the exam, which includes adequate time to complete the exam for non-native English speakers.

Exam Delivery

This is a proctored exam delivered through Pearson VUE. For more information, visit the Pearson VUE website.

Certification Information

For details and a complete list of requirements and recommendations for attainment, please reference the VMware Education Services – Certification website.

Minimally Qualified Candidate

The minimally qualified candidate (MQC) has both a strong conceptual understanding on Spring framework and programming experience. One way to prepare for the exam is to take 4-day Core Spring training.

Exam Sections

The exam is made of the following sections below.

- Section 1-1 – Container, Dependency, and IOC
- Section 1-2 – Aspect-Oriented Programming (AOC)
- Section 1-3 – Data Management: JDBC, Transactions
- Section 1-4 – Spring Data JPA
- Section 1-5 – Spring MVC Basics
- Section 1-6 – Spring MVC REST
- Section 1-7 – Security
- Section 1-8 – Testing
- Section 1-9 – Spring Boot Basics
- Section 1-10 – Spring Boot Auto Configuration
- Section 1-11 – Spring Boot Actuator
- Section 1-12 – Spring Boot Testing

Sections Included in this Exam

Section 1-1 – Container, Dependency and IOC

- What is dependency injection and what are the advantages of using it?
- What is an interface and what are the advantages of making use of them in Java?
- What is an ApplicationContext?
- How are you going to create a new instance of an ApplicationContext?
- Can you describe the lifecycle of a Spring Bean in an ApplicationContext?
- How are you going to create an ApplicationContext in an integration test?
- What is the preferred way to close an application context? Does Spring Boot do this for you?
- Are beans lazily or eagerly instantiated by default? How do you alter this behavior?
- What is a property source? How would you use @PropertySource?
- What is a BeanFactoryPostProcessor and what is it used for? When is it invoked?
- What is a BeanPostProcessor and how is it different to a BeanFactoryPostProcessor? What do they do? When are they called?
- What does component-scanning do?
- What is the behavior of the annotation @Autowired with regards to field injection, constructor injection and method injection?
- How does the @Qualifier annotation complement the use of @Autowired?
- What is a proxy object and what are the two different types of proxies Spring can create?
- What does the @Bean annotation do?
- What is the default bean id if you only use @Bean? How can you override this?
- Why are you not allowed to annotate a final class with @Configuration
- How do you configure profiles? What are possible use cases where they might be useful?
- Can you use @Bean together with @Profile?
- Can you use @Component together with @Profile?
- How many profiles can you have?
- How do you inject scalar/literal values into Spring beans?
- What is Spring Expression Language (SpEL for short)?
- What is the Environment abstraction in Spring?
- Where can properties in the environment come from – there are many sources for properties – check the documentation if not sure. Spring Boot adds even more.
- What can you reference using SpEL?
- What is the difference between $ and # in @Value expressions?

Section 1-2 – Aspect-Oriented Programming (AOP)

- What is the concept of AOP? Which problem does it solve? What is a cross cutting concern?
- What is a pointcut, a join point, an advice, an aspect, weaving?

- How does Spring solve (implement) a cross cutting concern?
- Which are the limitations of the two proxy-types?
- How many advice types does Spring support? Can you name each one?
- If shown pointcut expressions, would you understand them?
- What is the JoinPoint argument used for?
- What is a ProceedingJoinPoint? Which advice type is it used with?

Section 1-3 – Data Management: JDBC, Transactions

- What is the difference between checked and unchecked exceptions?
- How do you configure a DataSource in Spring?
- What is the Template design pattern and what is the JDBC template?
- What is a callback? What are the JdbcTemplate callback interfaces that can be used with queries? What is each 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).
- Can you execute a plain SQL statement with the JDBC template?
- When does the JDBC template acquire (and release) a connection, for every method called or once per template? Why?
- How does the JdbcTemplate support queries? How does it return objects and lists/maps of objects?
- What is a transaction? What is the difference between a local and a global transaction?
- Is a transaction a cross cutting concern? How is it implemented by Spring?
- How are you going to define a transaction in Spring?
- Is the JDBC template able to participate in an existing transaction?
- What is @EnableTransactionManagement for?
- How does transaction propagation work?
- What happens if one @Transactional annotated method is calling another @Transactional annotated method inside a same object instance?
- Where can the @Transactional annotation be used? What is a typical usage if you put it at class level?
- What does declarative transaction management mean?
- What is the default rollback policy? How can you override it?
- What is the default rollback policy in a JUnit test, when you use the @ RunWith(SpringJUnit4ClassRunner.class) in JUnit 4 or @ExtendWith(SpringExtension. class) in JUnit 5, and annotate your @Test annotated method with @Transactional?
- Are you able to participate in a given transaction in Spring while working with JPA?
- Which PlatformTransactionManager(s) can you use with JPA?
- What do you have to configure to use JPA with Spring? How does Spring Boot make this easier?

Section 1-4 – Spring Data JPA

- What is a Spring Data Repository interface?
- How do you define a Spring Data Repository interface? Why is it an interface not a class?
- What is the naming convention for finder methods in a Spring Data Repository interface?

- How are Spring Data repositories implemented by Spring at runtime?
- What is @Query used for?

Section 1-5 – Spring MVC Basics

- What is the @Controller annotation used for?
- How is an incoming request mapped to a controller and mapped to a method?
- What is the difference between @RequestMapping and @GetMapping?
- What is @RequestParam used for?
- What are the differences between @RequestParam and @PathVariable?
- What are the ready-to-use argument types you can use in a controller method?
- What are some of the valid return types of a controller method?

Section 6 – Spring MVC REST

- What does REST stand for?
- What is a resource?
- Is REST secure? What can you do to secure it?
- Is REST scalable and/or interoperable?
- Which HTTP methods does REST use?
- What is an HttpMessageConverter?
- Is @Controller a stereotype? Is @RestController a stereotype?
- What is the difference between @Controller and @RestController?
- When do you need to use @ResponseBody?
- What are the HTTP status return codes for a successful GET, POST, PUT or DELETE operation?
- When do you need to use @ResponseStatus?
- Where do you need to use @ResponseBody? What about @RequestBody?
- If you saw example Controller code, would you understand what it is doing? Could you tell if it was annotated correctly?
- What Spring Boot starter would you use for a Spring REST application?
- If you saw an example using RestTemplate, would you understand what it is doing?

Section 1-7 – Security

- What are authentication and authorization? Which must come first?
- Is security a cross cutting concern? How is it implemented internally?
- What is the delegating filter proxy?
- What is the security filter chain?
- What is a security context?
- What does the ** pattern in an antMatcher or mvcMatcher do?

- Why is the usage of mvcMatcher recommended over antMatcher?
- Does Spring Security support password encoding?
- 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).
- What do @PreAuthorized and @RolesAllowed do? What is the difference between them?
- How are these annotations implemented?
- In which security annotation, are you allowed to use SpEL?

Section 8 – Testing

- What type of tests typically use Spring?
- How can you create a shared application context in a JUnit integration test?
- When and where do you use @Transactional in testing?
- How are mock frameworks such as Mockito or EasyMock used?
- How is @ContextConfiguration used?
- How does Spring Boot simplify writing tests?
- What does @SpringBootTest do? How does it interact with @SpringBootApplication and @SpringBootConfiguration?

Section 1-9 – Spring Boot Basics

- What is Spring Boot?
- What are the advantages of using Spring Boot?
- What things affect what Spring Boot sets up?
- What is a Spring Boot starter? Why is it useful?
- Spring Boot supports both properties and YML files. Would you recognize and understand them if you saw them?
- Can you control logging with Spring Boot? How?
- Where does Spring Boot look for application.properties file by default?
- How do you define profile specific property files?
- How do you access the properties defined in the property files?
- What properties do you have to define in order to configure external MySQL?
- How do you configure default schema and initial data?
- What is a fat jar? How is it different from the original jar?
- What embedded containers does Spring Boot support?

Section 1-10 – Spring Boot Auto Configuration

- How does Spring Boot know what to configure?
- What does @EnableAutoConfiguration do?
- What does @SpringBootApplication do?

- Does Spring Boot do component scanning? Where does it look by default?
- How are DataSource and JdbcTemplate auto-configured?
- What is spring.factories file for?
- How do you customize Spring Boot auto configuration?
- What are the examples of @Conditional annotations? How are they used?

Section 1-11 – Spring Boot Actuator

- What value does Spring Boot Actuator provide?
- What are the two protocols you can use to access actuator endpoints?
- What are the actuator endpoints that are provided out of the box?
- What is info endpoint for? How do you supply data?
- How do you change logging level of a package using loggers endpoint?
- How do you access an endpoint using a tag?
- What is metrics for?
- How do you create a custom metric?
- What is Health Indicator?
- What are the Health Indicators that are provided out of the box?
- What is the Health Indicator status?
- What are the Health Indicator statuses that are provided out of the box?
- How do you change the Health Indicator status severity order?
- Why do you want to leverage 3rd-party external monitoring system?

Section 1-12 – Spring Boot Testing

- When do you want to use @SpringBootTest annotation?
- What does @SpringBootTest auto-configure?
- What dependencies does spring-boot-starter-test brings to the classpath?
- How do you perform integration testing with @SpringBootTest for a web application?
- When do you want to use @WebMvcTest? What does it auto-configure?
- What are the differences between @MockBean and @Mock?
- When do you want @DataJpaTest for? What does it auto-configure?

Recommended Courses

Core Spring 4-day Course

Spring Boot 2-day Course (If you are already familiar with Spring Framework)

References*

In addition to the recommended courses, item writers use the following references for information when writing exam questions. It is recommended that you study the reference content as you prepare to take the exam, in addition to any recommended training.

Name	Version
Spring Framework Core Technologies	Spring Framework 5.x
Spring Framework Data Access	Spring Framework 5.x
Spring Framework Testing	Spring Framework 5.x
Spring Framework MVC	Spring Framework 5.x
Spring Security	Spring Framework 5.x
Using Spring Boot	Spring Boot 2.3.x
Spring Boot Features	Spring Boot 2.3.x
Spring Boot Actuator	Spring Boot 2.3.x

Certification Requirements

PIV-SPC 2021

Sample Questions

Sample questions presented here are examples of the types of questions candidates may encounter and should not be used as a resource for exam preparation.

Sample Question 1

Which of the following statements describe the ApplicationContext object correctly (select two)

A. Statement #1
B. Statement #2
C. Statement #3
D. Statement #4

Sample Question 2

Which of the following statements best describe the “After Returning” advice type in Spring AOP? (select one)

A. Statement #1
B. Statement #2
C. Statement #3
D. Statement #4