Pivotal

Core Spring 4.2 Study Guide

COMPLETED: NOVEMBER 2015

Table of Contents

Overview

This guide is designed to help you prepare for the Core Spring 4.2 certification exam. Please be aware that it should not be used if you have attended a Core-Spring course that was using a previous version – they have their own dedicated certification guides.

At this time previous versions are:

- CoreSpringV4.0 based on Spring 4.0
- CoreSpringV4.1 based on Spring 4.1
- CoreSpringV3.2 based on Spring 3.2

The certification exam is based on the 4-day Core Spring training and the materials provided with it are the ideal source to use for preparation. Of course as with any certification the most valuable part, besides recognition, is the learning process. Hence we encourage you to take time to experiment and follow your curiosity when questions arise.

A 4-day course contains a lot of material. To help you focus your efforts and to know when you're ready we've put together this guide. The guide contains a list of topics and a list of further resources. Topics are organized by subject area, where each topic contains a description of what you should make sure you know.

The list of topics can be used as a check-list. The training materials can be used as a point of reference and as a learning ground. The list of resources is where you can go further for getting answers. Everything in the exam is covered somewhere in the course notes.
One possible way to prepare is to do the following for a given training module:

1. Review the slides, making notes of questions
2. Work through the lab
3. Review the list of topics that matches to the module by subject area
4. Use the lab to experiment with anything you need to spend more time on
5. Use the provided list of resources to look for further answers
6. Reading (at least partially) the reference documentation
7. Memorize the "big pictures", tables, overviews, etc

Of course there are many more ways to organize your efforts. You can pair up with someone else planning to take the exam or review all presentations for a given subject area before going through the labs. Or maybe you have access to actual applications you can review to test your knowledge.

Please keep in mind that you are expected to have good working knowledge of all the topics listed. Most of the questions will be very general, however you will be asked a few advanced questions.

Logistics

The certification can be done in any Pearson Vue test center – see this link:

http://home.pearsonvue.com/test-taker.aspx

Using the “a to z list of all programs” find “Pivotal”. Find a test center near you to make an appointment in advance and provide the voucher you have received via email. If you did not receive a voucher (may take up to 2 weeks) please email to eduction@pivotal.io.

Arrive at the test center early to reduce your stress. You should at least have one identity card with you (please read the instructions carefully which you have received from the test center). You put all your belongings into a locker - you are not allowed to have books, pencils, paper, mobile phone or any other electronic devices with you. The room in which you are doing the exam is usually under camera surveillance.

The Exam

The exam itself is a computer-based exam. The exam software first gives you some general instructions: how to navigate, how to mark a question, and so forth - please read it carefully.

Once you have agreed that you want to start, you have 90 minutes to answer 50 multiple-choice questions. You must answer 38 questions correctly (76%) in order to pass the exam.

Basic exam technique applies: read each question *carefully* and answer the question that was asked not what you thought was asked.

Exam FAQ

1. Is there anything in the exam, which was not covered in the course?
   No.

2. Do I have to know class names and method signatures?
   No. We think that this is why you are using an IDE - for us it's much more important that you've understood the concepts rather than learning method signatures.

3. Do I have to write, complete or rearrange source code?
   No. The only thing you should be able to do is read a snippet of code and understand what it's doing. This might be an example of a class implementing a listener and you will then see a couple of related questions. We do not ask you questions on things an IDE can do for you, like checking if the code will compile.

4. Do I have to know any other APIs like AspectJ-Expression-Language in Detail?
   No. Of course you should be able to read, understand and use AspectJ-Expression-Language (pointcut expressions) wherever it is necessary to configure Spring but this is not an exam about AspectJ.

5. How long is the voucher valid?
   The voucher is valid one year (please also check the expiry date in the email) and it allows you to do one attempt. If you don't pass , then you must purchase another voucher (not the whole course again) for yourself. Also please note - if you fail an attempt - there is a mandatory 6 weeks lockout period until you can have your next try.

6. Are the advanced slides part of the exam?
   No. Only the content presented before each chapter lab slide will be on the exam. Any course content presented after the chapter lab will not be on the exam. No content from the optional chapters will be on the exam.

Topics

The following is a list of topics, each of which is likely to have questions on the exam. Several of the bullet points below overlap, asking the same (or a related) question in a different way. The answer to more than one bullet point question may be the *same* as the answer to another question just before or after. *Don't let this confuse you*.

CONTAINER, DEPENDENCY, AND IOC

- What is dependency injection and what are the advantages?
- What is an interface and what are the advantages of making use of them in Java?
- What is meant by “application-context” and how do you create one?
- What is the concept of a “container” and what is its lifecycle?
- Dependency injection using Java configuration
- Dependency injection in XML, using constructor or setter injection
- Dependency injection using annotations (`@Component`, `@Autowired`)
- Component scanning, Stereotypes and Meta-Annotations
- Scopes for Spring beans. What is the default?
- What is an initialization method and how is it declared in a Spring bean?
- What is a destroy method, how is it declared and when is it called?
- What is a `BeanFactoryPostProcessor` and what is it used for?
- What is a `BeanPostProcessor` and how is the difference to a `BeanFactoryPostProcessor`? What do they do? When are they called?
- Are beans lazily or eagerly instantiated by default? How do you alter this behavior?
- 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 the role of the `@PostConstruct` and `@PreDestroy` annotations? When will they get called?
- What is a proxy object and what are the two different types of proxies Spring can create?
- What is the power of a proxy object and where are the disadvantages?
- What are the limitations of these proxies (per type)?
- How do you inject scalar/literal values into Spring beans?
- How are you going to create a new instance of an ApplicationContext?
- What is a prefix?
- What is the lifecycle on an ApplicationContext?
- What does the "`@Bean` annotation do?
- How are you going to create an ApplicationContext in an integration test or a *JUnit* test?
- What do you have to do, if you would like to inject something into a private field?
- What are the advantages of JavaConfig? What are the limitations?

- What is the default bean id if you only use "`@Bean`"?
- Can you use `@Bean` together with `@Profile`?
- What is Spring Expression Language (SpEL for short)?
- What is the environment abstraction in Spring?
- What can you reference using SpEL?
- How do you configure a profile. What are possible use cases where they might be useful?
- How many profiles can you have?
- How do you enable JSR-250 annotations like `@PostConstruct`?
- Why are you not allowed to annotate a final class with `@Configuration`
- Why must you have a default constructor in your `@Configuration` annotated class?
- Why are you not allowed to annotate final methods with `@Bean`?
- What is the preferred way to close an application context?
- How can you create a shared application context in a JUnit test?
- What does a static `@Bean` method do?
- What is a `ProperyPlaceholderConfigurer` used for?
- What is `@Value` used for?
- What is the difference between $ and # in `@Value` expressions?

ASPECT ORIENTED PROGRAMMING

- What is the concept of AOP? Which problem does it solve?
- 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. What are they used for?
- What do you have to do to enable the detection of the `@Aspect` annotation?
- Name three typical cross cutting concerns.
- What two problems arise if you don't solve a cross cutting concern via AOP?
- What does `@EnableAspectJAutoProxy` do?
- What is a named pointcut?
- How do you externalize pointcuts? What is the advantage of doing this?
- What is the `JoinPoint` argument used for?
- What is a `ProceedingJoinPoint`?
- What are the five advice types called?
- Which advice do you have to use if you would like to try and catch exceptions?

JDBC, TRANSACTIONS, AND ORM

- What is the difference between checked and unchecked exceptions?
- Why do we (in Spring) prefer unchecked exceptions?
- What is the data access exception hierarchy?

- How do you configure a DataSource in Spring? Which bean is very useful for development/test databases?
- What is the Template design pattern and what is the JDBC template?
- 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).
- Can you execute a plain SQL statement with the JDBC template?
- Does the JDBC template acquire (and release) a connection for every method called or once per template?
- Is the JDBC template able to participate in an existing transaction?
- 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 in Spring?
- How are you going to set up a transaction in Spring?
- What does @Transactional do? What is the `PlatformTransactionManager`?
- What is the `TransactionTemplate`? Why would you use it?
- What is a transaction isolation level? How many do we have and how are they ordered?
- How does the `JdbcTemplate` support generic queries? How does it return objects and lists/maps of objects?
- What does transaction propagation mean?
- What happens if one `@Transactional` annotated method is calling another `@Transactional` annotated method on the 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 `SpringJUnit4ClassRunner` and annotate your `@Test` annotated method with `@Transactional`?
- 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?
- 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?
- What do you need to do in Spring if you would like to work with JPA?
- Are you able to participate in a given transaction in Spring while working with JPA?
- What is the `PlatformTransactionManager`?
- What does `@PersistenceContext` do?
- What are disadvantages or ORM? What are the benefits?
- What is an "instant repository"? (*hint:* recall Spring Data)
- How do you define an “instant” repository?
- What is `@Query` used for?

SPRING MVC AND THE WEB PLAYER

- MVC is an abbreviation for a *design pattern*. What does it stand for and what is the idea behind it?
- Do you need spring-mvc.jar in your classpath or is it part of spring-core?
- What is the `DispatcherServlet` and what is it used for?
- Is the `DispatcherServlet` instantiated via an application context?
- What is the root application context? How is it loaded?
- What is the `@Controller` annotation used for? How can you create a controller without an annotation?
- What is the `ContextLoaderListener` and what does it do?
- What are you going to do in the web.xml. Where do you place it?
- How is an incoming request mapped to a controller and mapped to a method?
- What is the `@RequestParam` used for?
- What are the differences between `@RequestParam` and `@PathVariable`?
- What are some of the valid return types of a controller method?
- What is a `View` and what's the idea behind supporting different types of `View`?
- How is the right `View` chosen when it comes to the rendering phase?
- What is the `Model`?
- Why do you have access to the model in your `View`? Where does it come from?
- What is the purpose of the session scope?
- What is the default scope in the web context?
- Why are controllers testable artifacts?
- What does the `InternalResourceViewResolver` do?

SECURITY

- What is the delegating filter proxy?
- What is the security filter chain?
- In the notes several predefined filters were shown. Do you recall what they did and what order they occurred in?
- Are you able to add and/or replace individual filters?
- Is it enough to hide sections of my output (e.g. JSP-Page)?
- Why do you need the `intercept-url`?
- 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).
- Is security a cross cutting concern? How is it implemented internally?
- What do `@Secured` and `@RolesAllowed` do? What is the difference between them?
- What is a security context?
- In which order do you have to write multiple intercept-url's?
- How is a `Principal` defined?
- What is authentication and authorization? Which must come first?
- In which security annotation are you allowed to use SpEL?
- Does Spring Security support password hashing? What is salting?

REST

- What does REST stand for?
- What is a resource?
- What are safe REST operations?
- What are idempotent operations? Why is idempotency important?
- Is REST scalable and/or interoperable?
- What are the advantages of the RestTemplate?
- Which HTTP methods does REST use?
- What is an `HttpMessageConverter`?
- Is REST normally stateless?
- What does `@RequestMapping` do?
- Is `@Controller` a stereotype? Is `@RestController` a stereotype?
- What is the difference between `@Controller` and `@RestController`?
- When do you need `@ResponseBody`?
- What does `@PathVariable` do?
- What is the HTTP status return code for a successful DELETE statement?
- What does CRUD mean?
- Is REST secure? What can you do to secure it?
- Where do you need `@EnableWebMVC`?
- Name some common http response codes. When do you need `@ResponseStatus`?
- Does REST work with transport layer security (TLS)?
- Do you need Spring MVC in your classpath?

SPRING BOOT

- What is Spring Boot?
- What are the advantages of using Spring Boot?
- Why is it “opinionated”?
- How does it work? How does it know what to configure?
- What things affect what Spring Boot sets up?
- How are properties defined? Where?
- Would you recognize common Spring Boot annotations and configuration properties if you saw them in the exam?
- What is the difference between an embedded container and a WAR?
- What embedded containers does Spring Boot support?
- What does **`@EnableAutoConfiguration`** do? What about **`@SpringBootApplication`**?
- What is a Spring Boot starter POM? Why is it useful?
- Spring Boot supports both Java properties and YML files. Would you recognize and understand them if you saw them?
- Can you control logging with Spring Boot? How?
- Note that the second Spring Boot section (*Going Further*) is not required for this exam.

MICROSERVICES

- What is a microservices architecture?
- What are the advantages and disadvantages of microservices?
- What sub-projects of Spring Cloud did we cover in the course? Spring Cloud is a large umbrella project – only what we covered in the course will be tested.
- Would you recognize the Spring Cloud annotations and configuration we used in the course if you saw it in the exam?
- What Netflx projects did we use?
- How do you setup Service Discovery?
- How do you access a RESTful microservice?
- What is Eureka?

Resources

http://spring.io/blog
Blog: Point your favorite RSS reader or come back for detailed, quality posts by Spring developers.

http://docs.spring.io/spring/docs/current/spring-framework-reference
Reference: The reference documentation (800+ pages) is available as html pages, a single html page and as a PDF document.

http://docs.spring.io/spring/docs/current/javadoc-api
Javadoc API

http://springbyexample.org
Spring By Example: Another good repository with good code samples is *SpringByExample*.

Conclusion

When you worked through this guide and know all the answers, we are pretty confident that you should pass the certification. It's recommended to do it as soon as possible and we wish you good luck with it.

Thank you again for choosing Pivotal as your education partner and good luck with your projects.

If you have encountered any errors, have any suggestions or enquiries please don't hesitate to contact your trainer or write an email to eduction@pivotal.io.