Question #1 *Topic 1*

If a class is annotated with @Component, what should be done to have Spring automatically detect the annotated class and load it as a bean? (Choose the best answer.)

A. Ensure a valid bean name in the @Component annotation is specified.

B. Ensure a valid @ComponentScan annotation in the Java configuration is specified.

C. Ensure a valid @Scope for the class is specified.

D. Ensure a valid @Bean for the class is specified.

Correct Answer: *B*

Community vote distribution

B (100%)

faciorys Highly Voted 2 years ago

Selected Answer: B

To autodetect these classes and register the corresponding beans, you need to add @ComponentScan to your @Configuration class
https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#spring-core

upvoted 8 times

stefumies Most Recent 4 months, 2 weeks ago

B is correct and as per Spring documentation

upvoted 1 times

Evoila_TrainingMaterial 5 months, 2 weeks ago

Selected Answer: B

B. Ensure a valid @ComponentScan annotation in the Java configuration is specified.

This is the correct answer. The @ComponentScan annotation tells Spring where to look for classes annotated with @Component (and other stereotype annotations like @Service, @Repository, and @Controller). Without this, Spring will not know to scan the specified packages for annotated classes.

upvoted 2 times

Examtopics_admin 1 year ago

Test to be

upvoted 1 times

nesreenmhd123 1 year, 2 months ago

B is the correct answer: https://www.baeldung.com/spring-component-annotation

upvoted 2 times

Ancient1 1 year, 4 months ago

Selected Answer: B

In order for spring to find the beans, you need to add @ComponentScan

upvoted 1 times

zakupower 1 year, 10 months ago

Selected Answer: B

Valid bean name is provided for you using the name of the class, @ComponentScan on a @Configuration marked class is needed.

upvoted 3 times

Question #2 *Topic 1*

Which two options will inject the value of the daily.limit system property? (Choose two.)

A. @Value("#{daily.limit}")

B. @Value("$(systemProperties.daily.limit)")

C. @Value("$(daily.limit)")

D. @Value("#{systemProperties['daily.limit']}")

E. @Value("#{systemProperties.daily.limit}")

Correct Answer: *D*

Community vote distribution

CD (30%) | D (30%) | DE (20%) | Other

Uteman 2 months, 3 weeks ago

Selected Answer: E

Dand E are correct

upvoted 1 times

stefumies 4 months, 2 weeks ago

Of these answers onky D is correct it uses SPEL but with usual value injection it should be dollar sign and curly brace, not regular parenthisis

"#{systemProperties['daily.limit']}" is correct

Option C: @Value("$(daily.limit)") is BOT CORRECT it requires "{" not "("

upvoted 1 times

Evoila_TrainingMaterial 5 months, 2 weeks ago

Selected Answer: CD

Correct Choices:

Option C: @Value("${daily.limit}")

Option D: @Value("#{systemProperties['daily.limit']}")

These two options correctly inject the value of the daily.limit system property.

Incorrect Choices Explanation:

Option A: @Value("#{daily.limit}")

This syntax is incorrect because it looks for a bean named daily.limit in the Spring context, not a system property.

Option B: @Value("$(systemProperties.daily.limit)")

This is incorrect because the $(...) syntax is not valid in Spring's @Value annotation.

Option E: @Value("#{systemProperties.daily.limit}")

This is incorrect because the correct SpEL syntax requires the use of ['...'] for accessing properties dynamically.

upvoted 1 times

stefumies 2 months, 2 weeks ago

C. @Value("$(daily.limit)") is INCCORECT as it is using "(" parens not "{" curlies.

Im not sure if this is merely a font issue here, but it is not correct in this format. SPel uses curlies.

upvoted 1 times

Evoila_TrainingMaterial 5 months, 2 weeks ago

Correct Choices:

Option C: @Value("${daily.limit}")

Option D: @Value("#{systemProperties['daily.limit']}")

These two options correctly inject the value of the daily.limit system property.

Incorrect Choices Explanation:

Option A: @Value("#{daily.limit}")

This syntax is incorrect because it looks for a bean named daily.limit in the Spring context, not a system property.

Option B: @Value("$(systemProperties.daily.limit)")

This is incorrect because the $(...) syntax is not valid in Spring's @Value annotation.

Option E: @Value("#{systemProperties.daily.limit}")

This is incorrect because the correct SpEL syntax requires the use of ['...'] for accessing properties dynamically.

upvoted 1 times

james2033 11 months, 1 week ago

Probably, the question is incorrect. Correct way of Spring Express language like this https://docs.spring.io/spring-framework/docs/3.0.x/reference/expressions.html

value="#{ systemProperties['user.region'] }" and/or

value="#{ numberGuess.randomNumber }" and/or

@Value("#{ systemProperties['user.region'] }")
private String defaultLocale;

upvoted 1 times

Glothan 1 year ago

Selected Answer: DE

The C is not for systemproperties but only for application properties. In the exam the "(" are replaced by "{".
Correct answers are DE

upvoted 2 times

Glothan 1 year ago

Selected Answer: BD

The C is not for systemproperties but only for application properties. In the exam the "(" are replaced by "{".
Correct answers are BD

upvoted 1 times

Azuni 1 year, 4 months ago

Selected Answer: CD

This question appeared on my exam. There is an error on the answers. Answer C is different than in the exam. The text in the exam is @Value("${daily.limit}").

upvoted 4 times

Tolo01 1 year, 5 months ago

Selected Answer: D

Only D is the correct answer

upvoted 4 times

Azuni 1 year, 4 months ago

I also realized this now. The way the question is formulated, B and C, has () and not {}. Only D will work. If the () in B and C were changed to {}, then C and D will be correct.

upvoted 1 times

qqoo 1 year, 6 months ago

Answer is CD

upvoted 2 times

Azuni 1 year, 5 months ago

I tested all scenarios on my own IDE and it confirms your answer that it is C and D.

upvoted 1 times

rhuanca 1 year, 9 months ago

A. @Value("#{daily.limit}")

E. @Value("#{systemProperties.daily.limit}") because we are talking about SystemProperty

upvoted 4 times

Question #3 *Topic 1*

Which two options are REST principles? (Choose two.)

A. RESTful applications use a stateless architecture.

B. RESTful application use HTTP headers and status codes as a contract with the clients.

C. RESTful applications cannot use caching.

D. RESTful application servers keep track of the client state.

E. RESTful applications favor tight coupling between the clients and the servers.

Correct Answer: *AB*

Community vote distribution

AB (100%)

james2033 Highly Voted 4 months, 2 weeks ago

Selected Answer: AB

The two options that are REST principles are:
A. RESTful applications use a stateless architecture.
This means that the server does not keep any client state between requests. Each request from the client to the server must contain all the information needed to understand and process the request.

B. RESTful application use HTTP headers and status codes as a contract with the clients.
HTTP headers and status codes are used to communicate the status of the request, any errors that occurred, and metadata about the returned resource. This is part of the uniform interface constraint of REST.

Options C, D, and E are not principles of REST. RESTful applications can use caching (option C is incorrect), do not keep track of client state (option D is incorrect), and favor loose coupling between clients and servers (option E is incorrect).
upvoted 5 times

Question #4 *Topic 1*

Which option is true about use of mocks in a Spring Boot web slice test? (Choose the best answer.)

A. Mocking a Spring Bean requires annotating it with @MockBean annotation.

B. If a Spring Bean already exists in the web slice test spring context, it cannot be mocked.

C. Mocks cannot be used in a Spring Boot web slice test.

D. Mocking a Spring Bean requires annotating it with @Mock annotation.

Correct Answer: *A*

quakquak3 2 weeks, 3 days ago

Selected Answer: A

@MockBean adds a mock bean to the application context whether a bean of this type would exist otherwise or not

upvoted 1 times

PRASAD180 2 months, 2 weeks ago

C is crt

upvoted 1 times

james2033 4 months, 2 weeks ago

The question may be incorrect. https://spring.io/blog/2016/08/30/custom-test-slice-with-spring-boot-1-4

upvoted 1 times

Question #5 *Topic 1*

Which two statements are true regarding Spring Security? (Choose two.)

A. Access control can be configured at the method level.

B. A special Java Authentication and Authorization Service (JAAS) policy file needs to be configured.

C. Authentication data can be accessed using a variety of different mechanisms, including databases and LDAP.

D. In the authorization configuration, the usage of permitAll () allows bypassing Spring security completely.

E. It provides a strict implementation of the Java EE Security specification.

Correct Answer: *AC*

Community vote distribution

AC (100%)

james2033 4 months, 2 weeks ago

Selected Answer: AC

A and C.

upvoted 2 times

Tolo01 11 months, 1 week ago

Selected Answer: AC

A and C are the best answer

upvoted 1 times

zakupower 1 year, 4 months ago

Selected Answer: AC

permitAll() does not bypass security entirely, it just allows access to the specified matcher, but other security related filters still run. e.g. cors, cqrs ...

upvoted 3 times

Azuni 10 months, 3 weeks ago

Correct. The only way to bypass Spring Security completely is to use a WebSecurityCustomizer with the .ignore() method.

upvoted 1 times

trungviettri 1 year, 5 months ago

A,C correct

upvoted 4 times

Question #6 *Topic 1*

Which two statements are true regarding a Spring Boot-based Spring MVC application? (Choose two.)

A. The default embedded servlet container can be replaced with Undertow.

B. Jetty is the default servlet container.

C. Spring Boot starts up an embedded servlet container by default.

D. The default port of the embedded servlet container is 8088.

E. Spring MVC starts up an in-memory database by default.

Correct Answer: *AC*

Community vote distribution

AC (89%) 11%

james2033 5 months ago

Selected Answer: AC

(A) Undertow for Spring Reactive https://undertow.io/ See https://springhow.com/spring-boot-undertow/

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
<exclusions>
<exclusion>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-tomcat</artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-undertow</artifactId>
</dependency>
```

(B) Jetty for Spring Reactive.

(C) wrong, it is 8080, not 8088

(D) true

(E) wrong, use in-memory with Spring Data JPA or Spring Security or declaring H2 database in pom.xml or build.gradle .

upvoted 2 times

james2033 5 months ago

Selected Answer: CD

(A) Undertow for Spring Reactive https://undertow.io/

(B) Jetty for Spring Reactive.

(C) true,

(D) true,

(E) wrong, use in-memory with Spring Data JPA or Spring Security or declaring H2 database in pom.xml or build.gradle .

upvoted 1 times

james2033 5 months ago

I have mistake, default port is 8080, not 8088, make D is wrong.
upvoted 1 times

Azuni 5 months ago
Selected Answer: AC
Went though the VMWare Spring Boot course again for Configuration of WAR and JAR deployments and it confirms what the commenters below said.
upvoted 1 times

zakupower 10 months ago
Selected Answer: AC
Default embedded servlet container is Tomcat. Jetty and Undertow can be used instead of it.
upvoted 3 times

qulyt 11 months, 3 weeks ago
Selected Answer: AC
A - https://docs.spring.io/spring-boot/docs/1.5.3.RELEASE/reference/html/howto-embedded-servlet-containers.html#:~:text=own%20JettyEmbeddedServletContainerFactory.-,73.13,-Use%20Undertow%20instead
upvoted 3 times

Question #7 *Topic 1*

Which two statements are true regarding Spring and Spring Boot Testing? (Choose two.)

A. EasyMock is supported out of the box.

B. @SpringBootTest or @SpringJUnitConfig can be used for creating an ApplicationContext.

C. Mockito spy is not supported in Spring Boot testing by default.

D. The spring-test dependency provides annotations such as @Mock and @MockBean.

E. Integration and slice testing are both supported.

Correct Answer: *BE*

Community vote distribution

BE (50%) BC (33%) C (17%)

2211094 6 months ago

BD is most accurate and correct answer. Because of E does not elaborate clearly how slice and integration tests differs in spring and spring boot.

upvoted 1 times

stefumies 2 months, 2 weeks ago

WebMvc is an example of a SliceTest as it is merely testing the web controller layer

upvoted 1 times

2211094 6 months, 3 weeks ago

Correct answer is B and E

upvoted 2 times

Azuni 1 year, 4 months ago

Selected Answer: BC

I have reviewed this question over again and it seems that B and C are the most likely of answers due by process of illumination.

A is incorrect for obvious reasons

B is correct because @SpringBootTest *OR* @SpringJUnitConfig can be used. I had to wrong in my comment below. That OR makes a big difference.

C is correct as Mockito Spy is not supported (only by Spring Framework), but Mockito SpyBean is supported by default.

D is incorrect for the same reason as specified in my comment below.

E is incorrect for the same reason as specified in my comment below.

upvoted 2 times

Azuni 1 year, 5 months ago

Selected Answer: C

It seems that C . It is a very tricky question. The Spring AND Spring Boot is where the trick lies.

A) INCORRECT: Of course EasyBox is a dependency that is not out of the box.

B) INCORRECT: If you consider Spring AND Spring Boot. @SpringBootTest and @SpringJUnitConfig both work in a Spring Boot application, but you cannot use @SpringBootTest in a plain Spring application.

C) CORRECT: Mockito spy does come with Spring Boot, but it isn't supported by defualt.

D) INCORRECT: spring-boot-starter-test does in fact provide annotations for @Mock and @MockBean, but spring-test by itself doesn't.

E) INCORRECT: Only integration testing are supported by both. Slice testing is only a concept in Spring Boot, not in plain Spring.

upvoted 1 times

qqoo 1 year, 6 months ago

Selected Answer: BE

D - it is not chosen because @Mock is commonly associated with other mocking frameworks like Mockito.

upvoted 4 times

Tolo01 1 year, 5 months ago

E is not correct

While integration testing and slice testing are both supported in Spring and Spring Boot, they are not both supported by default using a single testing annotation.

Integration Testing: Integration testing with Spring is supported using @SpringBootTest or @SpringJUnitConfig.

Slice Testing: Slice testing is performed using specialized annotations like @WebMvcTest, @DataJpaTest, etc. These annotations load only a part of the application context to focus

upvoted 1 times

rhuanca 1 year, 9 months ago

B and D

mockito is supported

upvoted 3 times

Question #8 Topic 1

Refer to the exhibit.

```
public class ClientServiceImpl implements ClientService{
     @Transactional(propagation=Propagation.REQUIRED)
     public void update() {
          update2();
     }
     @Transactional(propagation=Propagation.REQUIRES_NEW)
     public void update2() {
     }
}
```

Assume that the application is using Spring transaction management which uses Spring AOP internally.
Choose the statement that describes what is happening when the update1 method is called? (Choose the best answer.)

A. There are 2 transactions because REQUIRES_NEW always runs in a new transaction.

B. An exception is thrown as another transaction cannot be started within an existing transaction.

C. There is only one transaction because REQUIRES_NEW will use an active transaction if one already exists.

D. There is only one transaction initiated by update1() because the call to update2() does not go through the proxy.

Correct Answer: *D*

Community vote distribution

D (83%) A (17%)

stefumies 2 months, 1 week ago

This is a poor set of avaiable answers:

update() (there is no update1() method) will create a new transaction when executed, however update2() is REQUIRES_NEW which will suspend that transaction and create a new one,
until it has completed, and then return to the suspended transaction begun by update(), thus:

A is the most correct as there wuill be 2 transactions (even though one is suspended)
B this is false, new transactions can be started within existing ones
C is incorrect as it never uses an existing transaction (it suspends it)
D is incorrect because update 2 does begin a new transaction and is within a proxy

Therefore A is the most accurate in this context
upvoted 1 times

stefumies 2 months, 1 week ago

However a closer look at the call stack of the transactions, update2() is called within the scope (and thus the proxy) of update() and therefore, despite REQUIRES_NEW starting a new transaction and suspending the initial one, it is none the less in the same Proxy created by update(), and in this context D is most accurate!
upvoted 1 times

Evoila_TrainingMaterial 5 months, 1 week ago

Selected Answer: D

n Spring, transaction management relies on Spring AOP (Aspect-Oriented Programming), which means transactions are managed through proxies. For a new transaction (like one specified with REQUIRES_NEW) to be started within an existing transaction, the method must be called through a Spring proxy.

In the code provided, update1 calls update2 directly. This means the call does not go through the Spring proxy, and hence, the REQUIRES_NEW propagation on update2 is not recognized. As a result, there is only one transaction, which is the one initiated by update1.

If update2 were to be called in a way that went through the proxy, such as through another Spring-managed bean, a new transaction would indeed be started because of the REQUIRES_NEW propagation. However, the direct call prevents this from happening, leading to only one transaction.
upvoted 2 times

2211094 6 months ago

A is correct
upvoted 1 times

Ancient1 1 year, 4 months ago
Selected Answer: D
Answer is D.

Based on the lecture video from Vmware's training course (Spring framework essentials> Module 9 > Configure Transaction Propagation):
The first update method will create a transaction and run in a proxy, and when the second update method is called, it will be executed within the same transaction in the same proxy.
upvoted 4 times

Ancient1 1 year, 4 months ago
Now you might be asking, why would it execute within the same transaction if the propagation is REQUIRES_NEW? Wouldn't it create a new transaction?

Normally yes, however, In this case, the update2() method is being called in the same class as the first update(), making it an internal call.

When the first update method was called the @Transactional annotation passed the method execution to the interceptor, which created a proxy for the transaction (Spring AOP). When the Internal call to the second update2() method was executed, it couldn't be intercepted by a new interceptor, therefore, it couldn't create its own transactional proxy, which means it cannot create a new transaction.

If you don't understand this concept right away, don't worry. I had to review the video several times and I'm still trying to wrap my head around it.
upvoted 1 times

Azuni 1 year, 4 months ago
Same here. Luckily the instructor in the video did mention this and illustrated this in his slides 3:42 minutes into the video. Took me a while to filter though all the stuff.
upvoted 1 times

Azuni 1 year, 5 months ago
Selected Answer: A
The answer should be A. There should be two transactions. The first one will be suspended when the second one is called and once the second transaction is done, the first one will become active again.

Please refer to the Spring Docs on the topic of Transaction propagation: https://docs.spring.io/spring-framework/reference/data-access/transaction/declarative/tx-propagation.html
upvoted 1 times

Azuni 1 year, 5 months ago
UPDATE: I went though the VMWare Spring Framework Essentials lecture video where they used this exact example. The answer is indeed D. Feel to stupid for my original post. Apologies for the confusion.
upvoted 4 times

Verixas 1 year, 8 months ago
The option D is correct due to this article, where this case resolved - https://www.marcobehler.com/guides/spring-transaction-management-transactional-in-depth
upvoted 2 times

rhuanca 1 year, 9 months ago
I think is A correct answer

Option B is incorrect because an exception is not thrown when using REQUIRES_NEW propagation.
Option C is incorrect because REQUIRES_NEW will always create a new transaction even if an active transaction is present.
Option D is also incorrect because the call to update2() does go through the proxy and the transactional behavior will be applied.
upvoted 1 times

Question #9 *Topic 1*

Which two statements are true concerning constructor injection? (Choose two.)

A. If there is only one constructor the @Autowired annotation is not required.

B. Constructor injection only allows one value to be injected.

C. Constructor injection is preferred over field injection to support unit testing.

D. Construction injection can be used with multiple constructors without @Autowired annotation.

E. Field injection is preferred over constructor injection from a unit testing standpoint.

Correct Answer: *AC*

Community vote distribution

AC (100%)

Ancient1 Highly Voted 4 months, 3 weeks ago

Selected Answer: AC

A: @Autowired is not necessary if you have one constructor that instantiates the fields

C: Constructor injection is preferred over field injection, as it helps with unit testing. Most IDE's will warn you against field injection anyways.

upvoted 5 times

antpao86 Most Recent 2 months, 4 weeks ago

Field injection is ambiguous and doesn't really tell you which dependencies are mandatory.

upvoted 1 times

Eymet 3 months ago

Selected Answer: AC

See other comments why

upvoted 2 times

zakupower 10 months ago

Selected Answer: AC

Contructor injection is preferred because it allows for easier unit testing. If only one constructor is available it will be used to inject dependencies without the need of @Autowired annotation.

upvoted 4 times

faciorys 1 year ago

As of Spring Framework 4.3, an @Autowired annotation on such a constructor is no longer necessary if the target bean only defines one constructor to begin with.

Answer A and C

upvoted 3 times

Question #10 *Topic 1*

Given an ApplicationContext containing three bean definitions of type Foo with bean ids foo1, foo2, and foo3, which three @Autowired scenarios are valid and will allow the ApplicationContext to initialize successfully? (Choose three.)

A. @Autowired public void setFoo (Foo foo) {...}

B. @Autowired @Qualifier ("foo3") Foo foo;

C. @Autowired public void setFoo (@Qualifier ("foo1") Foo foo) {...}

D. @Autowired private Foo foo;

E. @Autowired private Foo foo2;

F. @Autowired public void setFoo(Foo foo2) {...}

Correct Answer: *BCE*

Community vote distribution

BCE (83%) BCF (17%)

Berny123 Highly Voted 1 year, 10 months ago

A and D is not correct at all. I defined 3 beans definitions with those ids and in my service class used setter injection or simple Foo foo definition of Autowired-> gives error:

Could not autowire. There is more than one bean of 'Foo' type.
Beans:
foo1 (MyConfiguration.java), foo2 (MyConfiguration.java), foo3 (MyConfiguration.java)

Working cases per single case: B,C,E,F
But not works all 4 together, C and F say the setFoo method name is already defined.

So B, E, F will work -> but will inject only 2 beans, foo3 and foo2 -> foo2 in 2 places, but is at the end the same bean defined once through setter injection and second time through autowired filed injection.

Or B, E, C will work -> will inject foo1, foo2, foo3

So the only correct answer to inject beans foo1,foo2,foo3 with working program is B, E, C.
upvoted 7 times

stefumies Most Recent 2 months, 1 week ago

A D and F will fail as they are ambiguous calls to Foo by type, when there is more than one bean of that type.
Thus BCE will work due to the use of the @Qualifier annotation specifying which Foo to use. @Qualifier can be used on constructors, setters and fields.
upvoted 1 times

Evoila_TrainingMaterial 4 months, 3 weeks ago

Selected Answer: BCE

in both option C and F, we are using the same method setFoo() to autowire the beans, which would lead to a method redeclaration error upon compilation.
upvoted 1 times

stefumies 2 months, 1 week ago

I don't believe these calls are all in the same class, I think these are options for the answer
upvoted 1 times

Eymet 1 year, 3 months ago

Selected Answer: BCE

Although for E field injection is not recommended, F will conflict with C because of two methods setFoo with the same signature. So BCE will initialize an ApplicationContext without any errors.
upvoted 4 times

Azuni 1 year, 5 months ago

Selected Answer: BCF

I believe it is BCF. You can interchange E and F, because they will do the same thing, so the comments below are not wrong either, but here is why I say BCF instead of BCE.

Do not perform field injection on a private field (as illustrated in E). It will work when you run the application as per normal, but you remote the ability to initialize that field it in a unit test. In the VMWare Spring Framework Essentials course on the topic of Annotation-based configurations, the lecturer warned not to do it.

upvoted 1 times

stefumies 2 months, 1 week ago

F will fail as it is ambigulously attempting to inject a Foo when there is more than one Foo, the parameter name is irrelevant, it can be anything, the Type (Foo) without qualification will cause the problem here.

upvoted 1 times

Azuni 1 year, 5 months ago

remove the ability to initialize

upvoted 1 times

zakupower 1 year, 10 months ago

Selected Answer: BCE

@Qualifier annotation is used to specify a bean id when injecting. @Qualifier can be used both on fields and method parameters. If it is a field injection the name of the field can qualify the bean.

upvoted 2 times

Berny123 1 year, 10 months ago

The very first injection was foo1 setter injection (@5293), after that foo3 and foo2 (the upper definition in class gets injected first, so in my code, I had defined the order of injections as follows: B, E, C, but setter injection (C) happens first, then B and E). There is no problem with the same "foo" variable name of two different beans in one class (C and B injections), as the variable scopes are different, and in practice, in case C the foo1 bean can be used and processed immediately (first) or reassigned to another variable name, therefore with this injection configuration the injection of all three beans foo1, foo2, foo3 is possible.

upvoted 1 times

Berny123 1 year, 10 months ago

Explanation of my previous comment/answer:

Spring injection happens first by type definition, but the type is the same in all answers, so it looks for @Primary annotation, which is not found, then for @Qualifier annotations, and the last lookup, if @Qualifier was not found, is the resolution by bean name. If not successful, after all, the exception is thrown.

In my demo app, when I run B,E,C, my break point first stops at setter method setFoo where I can see in debug already injected beans:

Foo@5293 - foo1 - C

Foo@5297 - foo3 - B

Foo@5298 - foo2 - E

upvoted 1 times

Question #11 *Topic 1*

Which dependency enables an automatic restart of the application as code is changed during development of a Spring boot configuration on a web application? (Choose the best answer.)

A. spring-boot-devtools

B. spring-boot-initializr

C. spring-boot-starter-devtools

D. spring-boot-restart

Correct Answer: *A*

Community vote distribution

A (100%)

james2033 2 months, 1 week ago

Selected Answer: A

```
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<optional>true</optional>
</dependency>
</dependencies>
```

https://docs.spring.io/spring-boot/docs/1.5.16.RELEASE/reference/html/using-boot-devtools.html#using-boot-devtools

upvoted 3 times

rhuanca 9 months, 2 weeks ago

C is more specific for this purpose

upvoted 1 times

Verixas 5 months, 2 weeks ago

The is no spring-boot-starter-devtools dependency, A answer is correct (spring-boot-devtools)

upvoted 3 times

Question #12 *Topic 1*

Spring puts each bean instance in a scope. What is the default scope? (Choose the best answer.)

A. prototype

B. singleton

C. request

D. session

Correct Answer: *B*

Community vote distribution

B (100%)

james2033 2 months, 1 week ago

Selected Answer: B

The singleton scope is the default scope in Spring.

https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch04s04.html#:~:text=The%20singleton%20scope%20is%20the%20default%20scope%20in%20Spring.

upvoted 2 times

james2033 5 months ago

Selected Answer: B

Quote “The singleton scope is the default scope in Spring.” at https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch04s04.html at Section 4.4.1 , paragraph 3.

upvoted 2 times

saturnrings 5 months, 1 week ago

Obviously B, Singleton

upvoted 1 times

Question #13 *Topic 1*

Refer to the exhibit.

```
@PutMapping("/accounts/{id}")
public void update() {}
```

Which option is a valid way to retrieve the account id? (Choose the best answer.)

A. Add @PathVariable("id") String accountId argument to the update() handler method.

B. Add @PathVariable long accountId argument to the update() handler method.

C. Add @RequestParam long accountId argument to the update() handler method.

D. Add @RequestParam("id") String accountId argument to the update() handler method.

Correct Answer: *A*

Community vote distribution

A (100%)

Azuni Highly Voted 1 year, 4 months ago

Selected Answer: A

A is the correct answer.

Please refer to this: https://www.baeldung.com/spring-pathvariable#specifying-the-request-parameter-name

upvoted 7 times

2211094 Most Recent 6 months ago

Be careful here, to me this question is one of very tricky question. Answer is B and not A because A forces data type conversion from string to Integer/long.

upvoted 1 times

IYONISSIO 1 year, 4 months ago

The correct option is B. Add @PathVariable long accountId argument to the update() handler method.

The @PathVariable annotation is used to bind a path variable from the request URI to a parameter in the handler method. In this case, the path variable is id, and the type is long. This means that the account id will be passed to the update() handler method as a long value

upvoted 2 times

Azuni 1 year, 4 months ago

The answer is A. B will only be valid if the parameter name matches the path variable in the the URL.

upvoted 4 times

Question #14 *Topic 1*

Which strategy is correct for configuring Spring Security to intercept particular URLs? (Choose the best answer.)

A. The URLs can be specified via configuration (using authorizeRequests () and request matchers), with the most specific rule first and the least specific last.

B. Spring Security can obtain URLs from Spring MVC controllers, the Spring Security configuration just needs a reference to the controller to be protected.

C. The URLs are specified in a special properties file, used by Spring Security.

D. The URLs can be specified via configuration (using authorizeRequests () and request matchers), with the least specific rule first and the most specific last.

Correct Answer: *A*

Community vote distribution

A (80%) D (20%)

saJAva Highly Voted 6 months, 3 weeks ago

Selected Answer: A

it is important that more specific patterns are defined higher in the list than less specific patterns

upvoted 5 times

quakquak3 2 weeks, 3 days ago

yes, the rules for the first matched pattern are used

upvoted 1 times

softarts Most Recent 6 months, 1 week ago

A=>most specific rule first and the least specific last.

upvoted 2 times

james2033 11 months, 1 week ago

Selected Answer: D

Quote "A good practice is to define generic rules at the top and more specific rules at the bottom." at https://www.baeldung.com/spring-security-configuring-urls#1-allowing-requests-to-the-products-api

upvoted 1 times

Question #15 *Topic 1*

In which three ways are Security filters used in Spring Security? (Choose three.)

A. To provide risk governance.

B. To drive authentication.

C. To manage application users.

D. To provide a logout capability.

E. To enforce authorization (access control).

F. To encrypt data.

Correct Answer: *BDE*

Community vote distribution

BDE (100%)

saJAva 3 weeks, 2 days ago

Selected Answer: BDE

- A is incorrect.
- F is incorrect.

upvoted 3 times

james2033 4 months, 3 weeks ago

Selected Answer: BDE

- A is incorrect.
- F is incorrect.
- Spring security filter for authorization (What user can do what stuff), not authentication (Who are who?), therefore not for managing application users, causes C is incorrect.

upvoted 3 times

Question #16 *Topic 1*

Refer to the exhibit.

```
@Bean
@ConditionalOnBean (name= "dataSource")
public JdbcTemplate jdbcTemplate(DataSource dataSource) {
     return new JdbcTemplate(dataSource);
}
```

The above code shows a conditional @Bean method for the creation of a JdbcTemplate bean.
Which two statements correctly describe the code behavior? (Choose two.)

A. @ConditionalOnBean(name= “dataSource”) should be replaced with @ConditionalOnBean (DataSource.class) for greater flexibility.

B. @ConditionalOnBean(name= “dataSource”) should be replaced with @ConditionalOnMissingBean (DataSource.class) for greater flexibility.

C. The @Bean annotation should be removed.

D. A JdbcTemplate bean will be created when the DataSource class is in the classpath but there is no DataSource bean.

E. A JdbcTemplate bean will be created when a bean named dataSource has already been created.

Correct Answer: *AE*

Community vote distribution

AE (100%)

rhuanca Highly Voted 9 months, 2 weeks ago

A and E

Option D is incorrect because the @ConditionalOnBean annotation requires that a bean with the specified name or class already exists in the application context for the JdbcTemplate bean to be created.

upvoted 5 times

Eymet Most Recent 2 months, 3 weeks ago

Selected Answer: AE

Answer is AE

upvoted 4 times

qqoo 6 months ago

Selected Answer: AE

Answer is AE

upvoted 4 times

zakupower 10 months ago

Selected Answer: AE

These are correct

upvoted 4 times

Question #17 *Topic 1*

What is a Spring Boot starter dependency? (Choose the best answer.)

A. A setting for specifying which code you want Spring Boot to generate for you.

B. A specific POM which you must build to control Spring Boot's opinionated runtime.

C. A pre-existing model project you can download and use as the basis of your project.

D. An easy way to include multiple, coordinated dependencies related to a specific technology, like web or JDBC.

Correct Answer: *D*

Community vote distribution

D (100%)

james2033 Highly Voted 10 months, 3 weeks ago

Selected Answer: D

- A is incorrect.
- B is incorrect, not just runtime, starter need in compile time.
- C is incorrect, it is not an existing pom.xml for downloading.
- D is rationale. Let's see an example https://github.com/spring-projects/spring-boot/blob/main/spring-boot-project/spring-boot-starters/spring-boot-starter-web/build.gradle

upvoted 5 times

stefumies Most Recent 2 months, 1 week ago

D but terrible answer! its a preconfigured set of dependencies

upvoted 1 times

IYONISSIO 1 year, 4 months ago

A Spring Boot starter dependency is an easy way to include multiple, coordinated dependencies related to a specific technology, such as web or JDBC

upvoted 2 times

Question #18 *Topic 1*

Which two are required to use transactions in Spring? (Choose two.)

A. Add @EnableTransactionManagement to a Java configuration class.

B. Annotate a class, an interface, or individual methods requiring a transaction with the @Transactional annotation.

C. A class must be annotated with @Service and @Transaction.

D. A class requiring a transaction must implement the TransactionInterceptor interface.

E. Write a Spring AOP advice to implement transactional behavior.

Correct Answer: *AB*

Community vote distribution

AB (100%)

james2033 4 months, 3 weeks ago

Selected Answer: AB

Need 2 things: @EnableTransactionManagement and @Transactional .

upvoted 4 times

Ancient1 10 months, 4 weeks ago

Selected Answer: AB

A: You have to enable transaction management.

B: You choose which methods/classes will be treated as transactional by using the @Transactional annotation.

upvoted 4 times

Question #19 *Topic 1*

Which two statements are true regarding the RestTemplate class? (Choose two.)

A. It supports asynchronous non-blocking model.

B. It automatically supports sending and receiving Java objects.

C. It provides convenience methods for writing REST clients.

D. It provides convenience methods for writing REST services.

E. Sending an HTTP request with a custom header is not possible when using RestTemplate.

Correct Answer: *BC*

Community vote distribution

BC (100%)

Eymet 2 months, 3 weeks ago

Selected Answer: BC

B. It automatically supports sending and receiving Java objects.

True. RestTemplate is designed to work with Java objects and can automatically serialize and deserialize them to and from HTTP request and response bodies using message converters. You can configure it to work with different data formats such as JSON or XML.

C. It provides convenience methods for writing REST clients.

True. RestTemplate provides various convenience methods for making HTTP requests to RESTful services, including GET, POST, PUT, DELETE, etc. It simplifies the process of making REST calls.

upvoted 4 times

james2033 5 months ago

Selected Answer: BC

E is wrong. See https://www.baeldung.com/rest-template . Custom header RestTemplate https://stackoverflow.com/a/32623548/3728901

https://www.baeldung.com/rest-template#exchange_post-1

Admin please correct "Correct Answer".

upvoted 2 times

qqoo 6 months ago

Selected Answer: BC

Answer is BC

upvoted 2 times

zakupower 9 months, 3 weeks ago

Selected Answer: BC

RestTemplate supports sending custom headers using exchange method with HttpEntity object.

upvoted 2 times

Question #20 *Topic 1*

Which statement is true? (Choose the best answer.)

A. @ActiveProfiles is a class-level annotation that is used to instruct the Spring TestContext Framework to record all application events that are published in the ApplicationContext during the execution of a single test.

B. @ActiveProfiles is a class-level annotation that you can use to configure how the Spring TestContext Framework is bootstrapped.

C. @ActiveProfiles is a class-level annotation that you can use to configure the locations of properties files and inlined properties to be added to the set of PropertySources in the Environment for an ApplicationContext loaded for an integration test.

D. @ActiveProfiles is a class-level annotation that is used to declare which bean definition profiles should be active when loaded an ApplicationContext for an integration test.

Correct Answer: *D*

Community vote distribution

D (100%)

james2033 Highly Voted 10 months, 3 weeks ago

Selected Answer: D

'@ActiveProfiles is a class-level annotation that is used to declare which bean definition profiles should be active when loading an ApplicationContext for an integration test.'

Source https://docs.spring.io/spring-framework/reference/testing/annotations/integration-spring/annotation-activeprofiles.html

upvoted 5 times

2211094 Most Recent 6 months, 3 weeks ago

The Corect answer is D. And here is why others are not correct. A: @ActiveProfiles doesn't deal with recording application events. The Spring TestContext Framework handles event publication and listening separately.

B: @ActiveProfiles is specific to activating bean definition profiles for integration tests. It doesn't configure the overall bootstrapping of the TestContext Framework.

C: While @ActiveProfiles indirectly affects the environment, it doesn't directly configure the locations of property files or inline properties. That's typically done through @TestPropertySource or other configuration mechanisms.

upvoted 1 times

Question #21 *Topic 1*

Which two statements are true about REST? (Choose two.)

A. REST is a Protocol.

B. REST is Stateful.

C. REST is Reliable.

D. REST is Interoperable.

E. REST is Relative.

Correct Answer: *CD*

Community vote distribution

CD (91%) 9%

quakquak3 2 weeks, 3 days ago

Selected Answer: CD

https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm 5.1.3 states that being stateless makes REST reliable

upvoted 1 times

Evoila_TrainingMaterial 5 months ago

Selected Answer: DE

Interoperable: REST is a set of architectural principles that allows different systems to communicate over the web. It uses standard Http methods, which makes it highly interoperable across various platforms and technologies.

Relative: This might be a bit more nuanced, but in the context of REST, it could refer to the concept of HATEOAS (Hypermedia as the Engine of Application State), where resources contain links to other resources, making navigation and state transitions relative rather than absolute.

While REST can be used to build reliable systems, reliability is not an inherent property of REST itself; it depends on how the RESTful services are implemented.

upvoted 1 times

Azuni 1 year, 4 months ago

Selected Answer: CD

The best would be C and D, but I do believe the answer is only D, but using process of elimination, it can only be C and D. This is because A, B and E are all incorrect.

No where does the course material state that it is reliable. In fact, the reliability of a RESTful system depends on the underlying protocols and mechanisms used, such as HTTP's reliability mechanisms. REST itself doesn't guarantee reliability, but it can be implemented using reliable protocols.

upvoted 4 times

zakupower 1 year, 10 months ago

Selected Answer: CD

REST is a API Architectural Style not a Protocol

upvoted 3 times

faciorys 2 years ago

Selected Answer: CD

REST is not Protocol!

upvoted 3 times

faciorys 2 years ago

Answer C and D

REST is not Protocol!

upvoted 3 times

Question #22 *Topic 1*

Spring Boot will find and load property files in which of the following? (Choose the best answer.)

A. A *.properties file matching the name of the class annotated with @SpringBootApplication.

B. config.properties or config.yml, usually located in the classpath root.

C. application.properties or application.yml, usually located in the classpath root.

D. env.properties or env.yml, usually located in the classpath root.

Correct Answer: *C*

Community vote distribution

C (100%)

james2033 5 months ago

Selected Answer: C

Configuration file is application.properties or application.yml . Definitely.

upvoted 3 times

Question #23 *Topic 1*

Which three dependencies are provided by the spring-boot-starter-test? (Choose three.)

A. Cucumber

B. Hamcrest

C. spring-test

D. Junit

E. EasyMock

F. PowerMock

Correct Answer: *BCD*

Community vote distribution

BCD (100%)

NAZER123 3 months ago

Correct answer: B C D

upvoted 3 times

james2033 5 months ago

Selected Answer: BCD

The spring-boot-starter-test “Starter” (in the test scope) contains the following provided libraries:

JUnit 5: The de-facto standard for unit testing Java applications.

Spring Test & Spring Boot Test: Utilities and integration test support for Spring Boot applications.

AssertJ: A fluent assertion library.

Hamcrest: A library of matcher objects (also known as constraints or predicates).

Mockito: A Java mocking framework.

JSONassert: An assertion library for JSON.

JsonPath: XPath for JSON.

at https://docs.spring.io/spring-boot/docs/3.2.x/reference/htmlsingle/#features.testing.test-scope-dependencies

Correct answer: B C D

upvoted 4 times

Azuni 4 months, 2 weeks ago

Very good explanation. The very same was presented in the VMWare Spring Boot course.

upvoted 1 times

qqoo 6 months ago

Selected Answer: BCD

These are correct

upvoted 2 times

zakupower 9 months, 3 weeks ago

Selected Answer: BCD

These are correct

upvoted 2 times

ridasys 1 year, 1 month ago

Selected Answer: BCD

https://docs.spring.io/spring-boot/docs/1.5.7.RELEASE/reference/html/boot-features-testing.html

Test scope dependencies

If you use the spring-boot-starter-test 'Starter' (in the test scope), you will find the following provided libraries:

JUnit — The de-facto standard for unit testing Java applications.

Spring Test & Spring Boot Test — Utilities and integration test support for Spring Boot applications.

AssertJ — A fluent assertion library.

Hamcrest — A library of matcher objects (also known as constraints or predicates).

Mockito — A Java mocking framework.

JSONassert — An assertion library for JSON.

JsonPath — XPath for JSON.

upvoted 3 times

Question #24 *Topic 1*

Which two statements are correct regarding Spring Boot auto-configuration customization? (Choose two.)

A. Use the @AutoConfigureAfter or @AutoConfigureBefore annotations to apply configuration in a specific order.

B. Disable specific auto-configuration classes by using the exclude attribute on the @EnableAutoConfiguation annotation.

C. Provide customized auto-configuration by subclassing the provided Spring Boot auto-configuration classes.

D. Enable component scanning within auto-configuration classes to find necessary components.

E. Control the order of auto-configuration classes applied with @AutoConfigureOrder.

Correct Answer: *AB*

Community vote distribution

AB (100%)

Azuni 4 months, 2 weeks ago

Selected Answer: AB

I believe A and B is correct.
The statement in A is exactly extracted from here: https://docs.spring.io/spring-boot/docs/3.0.0/reference/html/features.html#features.developing-auto-configuration.locating-auto-configuration-candidates
B is correct as it is stated as one of the many ways to customize your auto-configuration.
C is not correct as you can't just create a subclass of the auto-configuration class and expect it to work. You need to import that parent auto-configuration into a configuration of your own.
D is incorrect as component scanning has nothing to do with this.
E is actually also correct.

We have to chose only 2, I looked at which statement between A and E were exactly the same as in the Spring Docs. A was exactly the same as in the documentation.

This is a very peculiar question as it's answers are not really in the studying material, except for answer B.
upvoted 3 times

quakquak3 2 weeks, 3 days ago

I don't see why E is not also correct. I would have voted ABE if it was possible.

If your configuration needs to be applied in a specific order, you can use the before, beforeName, after and afterName attributes on the @AutoConfiguration annotation or the dedicated @AutoConfigureBefore and @AutoConfigureAfter annotations. For example, if you provide web-specific configuration, your class may need to be applied after WebMvcAutoConfiguration.

https://docs.spring.io/spring-boot/docs/3.0.0/reference/html/features.html#features.developing-auto-configuration.locating-auto-configuration-candidates mentions both A and E:
"If you want to order certain auto-configurations that should not have any direct knowledge of each other, you can also use @AutoConfigureOrder. That annotation has the same semantic as the regular @Order annotation but provides a dedicated order for auto-configuration classes."
upvoted 1 times

rhuanca 9 months ago

B and C .

A. The @AutoConfigureAfter, @AutoConfigureBefore, and @AutoConfigureOrder annotations are used to control the order in which auto-configuration classes are applied, not to apply configuration in a specific order
upvoted 2 times

Azuni 4 months, 2 weeks ago

It can't be C. It is not enough to simply subclass an auto-configuration class. It needs to be the same type of BEAN as the auto-configured bean. Not just the class.

upvoted 1 times

Question #25 *Topic 1*

Which two statements about the @Autowired annotation are true? (Choose two.)

A. @Autowired fields are injected after any config methods are invoked.

B. Multiple arguments can be injected into a single method using @Autowired.

C. By default, if a dependency cannot be satisfied with @Autowired, Spring throws a RuntimeException.

D. If @Autowired is used on a class, field injection is automatically performed for all dependencies.

E. @Autowired can be used to inject references into BeanPostProcessor and BeanFactoryPostProcessor.

Correct Answer: *BC*

Community vote distribution

BC (88%) 13%

Azuni Highly Voted 5 months ago

Selected Answer: BC

B is of course correct, but the reason why C is correct is because the default required setting of Autowired is 'true'. If no bean is found, a org.springframework.beans.factory.NoSuchBeanDefinitionException is thrown, which is a RuntimeException.

upvoted 6 times

Azuni 4 months, 2 weeks ago

A is not correct as the Fields are injected right after construction of a bean, before any config methods are invoked.
D is incorrect as you can't annotate a class.
E is incorrect as Autowired is not supported in BeanPostProcessor or BeanFactoryPostProcessor, so you can't inject a reference in them using Autowired.

Here is my reference: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Autowired.html

upvoted 1 times

Tolo01 Most Recent 5 months, 1 week ago

Selected Answer: BC

B and C

upvoted 2 times

qqoo 6 months ago

Selected Answer: AB

The others seem wrong

upvoted 1 times

rhuanca 9 months ago

B and C

upvoted 1 times

qulyt 11 months, 3 weeks ago

A and B, per Autowired documentation.

Autowired Fields
Fields are injected right after construction of a bean, before any config methods are invoked. Such a config field does not have to be public.

Autowired Methods
Config methods may have an arbitrary name and any number of arguments; each of those arguments will be autowired with a matching bean in the Spring container. Bean property setter methods are effectively just a special case of such a general config method. Such config methods do not have to be public.

upvoted 1 times

rhuanca 9 months, 1 week ago

Not sure about A, because @Autowired fields are injected before any config methods (@Bean, @Component, @Configuration, etc..) are invoked.

upvoted 1 times

faciorys 1 year ago

E is not correct

Note that actual injection is performed through a BeanPostProcessor which in turn means that you cannot use @Autowired to inject references into BeanPostProcessor or BeanFactoryPostProcessor types. Please consult the javadoc for the AutowiredAnnotationBeanPostProcessor class (which, by default, checks for the presence of this annotation).

https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Autowired.html

upvoted 2 times

Question #26 Topic 1

Which two statements are correct regarding the @EnableAutoConfiguration annotation? (Choose two.)

A. It is a meta-annotation on the @SpringBootApplication composed annotation.

B. It enables auto-configuration of the ApplicationContext by attempting to guess necessary beans.

C. It is meta-annotation on the @SpringBootConfiguration composed annotation.

D. It has the same effect regardless of the package of the class that is annotated with it.

E. It ensures auto-configuration is applied before user-defined beans have been registered.

Correct Answer: *AB*

Community vote distribution

AB (100%)

faciorys Highly Voted 6 months, 3 weeks ago

Selected Answer: AB

Enable auto-configuration of the Spring Application Context, attempting to guess and configure beans that you are likely to need
https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/EnableAutoConfiguration.html

upvoted 7 times

Question #27 *Topic 1*

Which two statements are true concerning the BeanPostProcessor Extension point? (Choose two.)

A. BeanPostProcessors are called before the dependencies have been injected.

B. Custom BeanPostProcessrs can be implemented for Spring applications.

C. BeanPostProcessors are called before the BeanFactoryPostProcessors.

D. BeanPostProcessors are called during the initialization phase of a bean life cycle.

E. BeanPostProcessors cannot be ordered in a Spring Boot application.

Correct Answer: *BD*

Community vote distribution

BD (100%)

saJAva 3 weeks, 2 days ago

Selected Answer: BD

correct

upvoted 3 times

Question #28 *Topic 1*

Which two statements are true about @Controller annotated classes? (Choose two.)

A. The @Controller annotated classes can only render views.

B. The classes are eligible for handling requests in Spring MVC.

C. The classes must be annotated together with @EnableMvcMappings to be discovered via component scanning.

D. @Controller is interchangeable with @RestController with no extra code changes for the methods inside the class.

E. The @Controller annotation is a stereotype annotation like @Component.

Correct Answer: *BE*

Community vote distribution

BE (100%)

saJAva 3 weeks, 2 days ago

Selected Answer: BE

correct

upvoted 2 times

Question #29 *Topic 1*

Which three types can be used as @Controller method arguments? (Choose three.)

A. Locale

B. Principal

C. Language

D. Session

E. Request

F. HttpSession

Correct Answer: *ABF*

Community vote distribution

ABF (75%) ABE (25%)

2211094 6 months, 3 weeks ago

Correct answer is ABF. Locale - represent use's local such as language or country
Principal - represent currently logged/authenticated user
HttpSession - represent user's current HTTP Session

so request and Session are too general thus why they're not correct and Language is part of Locale, and spring does not know language but locale.

upvoted 1 times

2211094 6 months, 3 weeks ago

https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-controller/ann-methods/arguments.html

upvoted 1 times

saJAva 6 months, 3 weeks ago

Selected Answer: ABE

correct

upvoted 1 times

Tolo01 1 year, 5 months ago

Selected Answer: ABF

https://docs.spring.io/spring-framework/reference/web/webmvc/mvc-controller/ann-methods/arguments.html

upvoted 4 times

rhuanca 1 year, 9 months ago

A, B , F , Locale specify what session users want (depends on location or language) , httpSession is the session we use for client and server, and principal is user authentication

upvoted 2 times

Question #30 *Topic 1*

Which three statements are advantages of using Spring's Dependency Injection? (Choose three.)

A. Dependency injection can make code easier to trace because it couples behavior with construction.

B. Dependency injection reduces the start-up time of an application.

C. Dependencies between application components can be managed external to the components.

D. Configuration can be externalized and centralized in a small set of files.

E. Dependency injection creates tight coupling between components.

F. Dependency injection facilitates loose coupling between components.

Correct Answer: *CDF*

Community vote distribution

CDF (100%)

saJAva 3 weeks, 2 days ago

Selected Answer: CDF

correct

upvoted 3 times

Tolo01 11 months, 1 week ago

Selected Answer: CDF

I think C, D and F are the best answer

upvoted 3 times

rhuanca 1 year, 3 months ago

why A and C are wrong ?

upvoted 1 times

nesreenmhd123 9 months ago

A. is incorrect. Dependency Injection decouples behavior from construction, making it easier to manage and test components independently.

upvoted 1 times

Question #31 *Topic 1*

Which two statements are correct when @SpringBootApplication is annotated on a class? (Choose two.)

A. It causes Spring Boot to enable auto-configuration by default.

B. Component scanning will start from the package of the class.

C. All other annotations on the class will be ignored.

D. Methods in the class annotated with @Bean will be ignored.

E. A separate ApplicationContext will be created for each class annotated with @SpringBootApplication.

Correct Answer: *AB*

Community vote distribution

AB (100%)

2211094 6 months, 3 weeks ago

Correct answer is AB. A reason why E is not a valid answer is because in typical Apring boot application, there will only be one ApplicationContext created however you can have multiple but this will lead to complication and unexpected behaviour to happen.

upvoted 2 times

Tolo01 1 year, 5 months ago

Selected Answer: AB

A and B

upvoted 4 times

zakupower 1 year, 9 months ago

Selected Answer: AB

These are correct

upvoted 2 times

Question #32 *Topic 1*

Refer to the exhibit.

```
ClientService service = applicationContext.getBean (ClientService.class);
```

It is a Java code fragment from a Spring application. Which statement is true with regard to the above example? (Choose the best answer.)

A. This syntax is invalid because the result of the getBean() method call should be cast to ClientService.

B. It will return a bean called ClientService regardless of its id or name.

C. This syntax is invalid because the bean id must be specified as a method parameter.

D. It will return a bean of the type ClientService regardless of its id or name.

Correct Answer: *D*

quakquak3 2 weeks, 3 days ago

Selected Answer: D

The bean is requested by type and no casting is necessary

upvoted 1 times

Azuni 4 months, 3 weeks ago

The answer is correct. Just remember though that you can fetch it with bean name, class or both (which was a surprise to me).

https://www.baeldung.com/spring-getbean#3-retrieving-bean-by-type

upvoted 4 times

Question #33 *Topic 1*

Which two statements about pointcut expressions are true? (Choose two.)

A. A pointcut expression cannot specify the type of parameters.

B. A pointcut expression will throw an exception if no methods are matched.

C. A pointcut expression cannot have a wildcard for a method name.

D. A pointcut expression can include operators such as the following: && (and), || (or), ! (not).

E. A pointcut expression can be used to select join points which have been annotated with a specific annotation.

Correct Answer: *DE*

quakquak3 2 weeks, 3 days ago

Selected Answer: DE

A: Method argument types can be matched, see https://docs.spring.io/spring-framework/reference/core/aop/ataspectj/advice.html#aop-ataspectj-advice-params-passing
B: If no methods are matched, the aspect will simply not be executed
C: Wildcards are supported in method names, see https://docs.spring.io/spring-framework/reference/core/aop/ataspectj/pointcuts.html#aop-pointcuts-examples
D: logical operators are supported: https://docs.spring.io/spring-framework/reference/core/aop/ataspectj/pointcuts.html#aop-pointcuts-combining
E: annotations can be used for selection: https://docs.spring.io/spring-framework/reference/core/aop/ataspectj/pointcuts.html#aop-pointcuts-designators

upvoted 1 times

Azuni 4 months, 3 weeks ago

D and E is correct.

Refer to: https://docs.spring.io/spring-framework/reference/core/aop/ataspectj/pointcuts.html

upvoted 2 times

Question #34 *Topic 1*

Which three types of objects can be returned form a JdbcTemplate query? (Choose three.)

A. Generic MapS

B. Simple types (int, long, String, etc)

C. JSONObject

D. User defined types

E. Properties

F. XMLObject

Correct Answer: *ABD*

Community vote distribution

ABD (100%)

Evoila_TrainingMaterial 5 months, 1 week ago

Selected Answer: ABD

A. Generic Maps: JdbcTemplate can return results as a List<Map<String, Object>>, where each Map represents a row with column names as keys.

B. Simple types (int, long, String, etc): JdbcTemplate can return single values directly, such as an int, long, or String, especially useful for aggregate functions or queries expected to return a single value.

D. User defined types: JdbcTemplate can map query results to custom user-defined types (JavaBeans) using a RowMapper or BeanPropertyRowMapper.

upvoted 2 times

3cef6cd 8 months ago

Selected Answer: ABD

GenericMaps, SimpleObjects and User Defined Types are returned.

upvoted 3 times

Azuni 1 year, 4 months ago

Selected Answer: ABD

Generic Maps (Map<String, Object>) is one of the return types of the JdbcTemplate. To return Properties, you need to create a custom RowMapper to map the ResultSet to a Property. It doesn't support it by default.

upvoted 1 times

Azuni 1 year, 4 months ago

I went tough the official VMWare course work and they state exactly that GenericMaps, SimpleObjects and User Defined Types are returned.

upvoted 1 times

rhuanca 1 year, 9 months ago

why not A instead of E. Generic Maps are more common query than properties in JdbcTemplate.

upvoted 2 times

Question #35 *Topic 1*

Which two use cases can be addressed by the method level security annotation @PreAuthorize? (Choose two.)

A. Allow access to a method based on user identity.

B. Allow access to a method based on the returned object.

C. Allow access to a method based on HTTP method.

D. Allow access to a method based on request URL.

E. Allow access to a method based on roles.

Correct Answer: *AE*

Community vote distribution

AE (100%)

Tolo01 5 months, 1 week ago

Selected Answer: AE

A and E

upvoted 4 times

zakupower 9 months, 3 weeks ago

Selected Answer: AE

These are correct

upvoted 2 times

Question #36 *Topic 1*

Which statement is true about the @PropertySource annotation? (Choose the best answer.)

A. Used to designate the location of the application.properties file in a Spring Boot application.

B. Used to easily look up and return a single property value from some external property file.

C. Used to designate the file directory of the application.properties file in a Spring Boot application.

D. Used to add a set of name/value pairs to the Spring Environment from an external source.

Correct Answer: *D*

Community vote distribution

D (60%) A (40%)

nesreenmhd123 2 months, 3 weeks ago

D is correct.

https://docs.spring.io/spring-framework/reference/core/beans/environment.html#beans-using-propertysource

upvoted 4 times

Tolo01 5 months ago

Selected Answer: D

Option A is not correct: @PropertySource is not specifically used to designate the location of the application.properties file in a Spring Boot application. In a Spring Boot application, the application.properties file is automatically loaded from the classpath by default without needing to use @PropertySource.

Option B is not entirely accurate: While you can use @PropertySource to load properties from an external property file, it allows you to load multiple key-value pairs rather than just a single property value.

Option C is not correct: @PropertySource does not designate the file directory of the application.properties file in a Spring Boot application. Again, in Spring Boot, the application.properties file is automatically loaded from the classpath by default without needing to use @PropertySource.

upvoted 4 times

rhuanca 9 months, 1 week ago

why not D ? I can not see any problem with D

upvoted 1 times

zakupower 9 months, 3 weeks ago

Selected Answer: A

None are exactly correct. It is an annotation to be used in tests to specify properties files to be included. Also it overrides the standard application.properties file.

upvoted 2 times

Question #37 *Topic 1*

Which two options are valid optional attributes for Spring's @Transactional annotation? (Choose two.)

A. isolation

B. writeOnly

C. nestedTransaction

D. readWrite

E. propagation

Correct Answer: *AE*

Community vote distribution

AE (100%)

james2033 5 months ago

Selected Answer: AE

https://www.baeldung.com/transaction-configuration-with-jpa-and-spring

@Transactional(propagation = Propagation.SUPPORTS, readOnly = true)

https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/transaction/annotation/Transactional.html

upvoted 3 times

Question #38 *Topic 1*

Which two statements are true about Spring Boot and Spring Data JPA? (Choose two.)

A. @EntityScan and spring.jpa.* properties can be used to customize Spring Data JPA.

B. Any kind of Hibernate property can be passed to Spring Data JPA like spring.jpa.properties.xxx.

C. Spring Data JPA is the only implementation for relational databases.

D. Scanning of JPA Entities can not be customized, the whole classpath is scanned.

E. Embedded Databases (H2, HSQLDB, Derby) are not re-created during the startup.

Correct Answer: *AB*

Community vote distribution

AB (80%) BE (20%)

Tolo01 Highly Voted 11 months, 1 week ago

Selected Answer: AB

These are correct

upvoted 5 times

saJAva Most Recent 3 weeks, 2 days ago

Selected Answer: BE

correct

upvoted 1 times

rhuanca 1 year, 3 months ago

hy not A and B .

C. not sure but it is possible with additional modules work with different Data bases non relational like MongoDB

E. Embedded databases (H2, HSQLDB, Derby) are re-created during the startup by default. However, this behavior can be controlled using the spring.datasource.initialization-mode property.

upvoted 1 times

Question #39 *Topic 1*

Refer to the exhibit.

```
@Configuration
public class MyConfig {
     @Bean
     public AccountRepository accountRepository() {
          return new JdbcAccountRepository() ;
     }
     @Bean
     public TransferService transferService() {
          TransferServiceImpl service = new TransferServiceImpl() ;
          service.setAccountRepository(accountRepository());
          return service;
     }
     @Bean
     public AccountService accountService() {
          return new AccountServiceImpl(accountRepository());
     }
}
```

Based on the default Spring behavior, choose the correct answer. (Choose the best answer.)

A. One AccountRepository bean will be instantiated since the default scope is singleton.

B. Three AccountRepository beans will be instantiated as the accountRepository() method will be called three times.

C. Many AccountRepository beans will be instantiated, depending how often accountRepository(), transferService() and accountService() are called.

D. Two AccountRepository beans will be instantiated as the accountRepository() method will be called two times.

Correct Answer: *A*

Community vote distribution

A (100%)

Tolo01 5 months ago

Selected Answer: A

A is the best answer

upvoted 2 times

rhuanca 9 months, 1 week ago

why not D ?

even if it is singleton because ther are no scope , the bean is still instantiated twice because it is called twice by two different @Bean methods (transferService() and accountService()).

upvoted 1 times

nesreenmhd123 2 months, 3 weeks ago

A. One AccountRepository bean will be instantiated since the default scope is singleton.

In Spring, when a method annotated with @Bean is called multiple times within the same application context, the Spring container manages the bean instances based on the method's scope. By default, the scope of a @Bean method is singleton, which means Spring will create and manage a single instance of the bean within the application context. Regardless of how many times the method is called, only one instance of the bean will be created and managed by Spring.

upvoted 3 times

zakupower 9 months, 3 weeks ago

Selected Answer: A

Default scope is singleton, doesn't matter how many times you request the bean.

upvoted 3 times

quakquak3 2 weeks, 3 days ago

Also, Spring intercepts the call to the bean factory method and first checks if the singleton is already created. So it makes no difference if accountRepository() is called or if a parameter of type AccountRepository is used in transferService()/accountService()

upvoted 1 times

Question #40 *Topic 1*

Refer to the exhibit.

```
@Configuration
@ConditionalOnClass (HelloService.class)
public class HelloAutoConfig {
     @ConditionalOnMissingBean(HelloService.class)
     @Bean
     HelloService helloService() {
          return new TypicalHelloService ();
     }
}
```

Which two statements are correct regarding the HelloAutoConfig auto-configuration class when it is specified in the META-INF/spring.factories file? (Choose two.)

A. A HelloService bean will be created from the helloService() method even if the HelloService.class is not in the classpath.

B. A HelloService bean will be created from the helloService() method only when there is no other HelloService bean in the ApplicationContext.

C. This auto-configuration class is used only when the HelloService.class is not on the classpath.

D. This auto-configuration class is used only when the HelloService.class is on the classpath.

E. A HelloService bean will be created from the helloService() method and will replace existing a HelloService bean in the ApplicationContext.

Correct Answer: *BD*

quakquak3 2 weeks, 3 days ago

Selected Answer: BD

@ConditionalOnClass means that the specified class has to be on the classpath for the configuration to be applied
@ConditionalOnMissingBean means that the bean definition is only used if there is not already a bean of the given class or given name defined (manually by @Component or @Bean or by another autoconfiguration applied before)

upvoted 1 times

2211094 6 months ago

Correct is BC

upvoted 2 times

Azuni 1 year, 4 months ago

The answers are correct.
Refer to:
1) https://docs.spring.io/spring-boot/docs/1.2.5.RELEASE/reference/html/boot-features-developing-auto-configuration.html#boot-features-class-conditions
2) https://docs.spring.io/spring-boot/docs/1.2.5.RELEASE/reference/html/boot-features-developing-auto-configuration.html#boot-features-bean-conditions

upvoted 2 times

Question #41 *Topic 1*

Refer to the exhibit.

```
@Configuration
public class AppConfig {
      @Bean
      public ClientService clientService() {
            return new ClientServiceImpl();
      }
}
```

What is the id/name of the declared bean in this Java configuration class? (Choose the best answer.)

A. clientServiceImpl (starting with lowercase “c”)

B. clientServiceImpl (starting with uppercase “C”)

C. clientService (starting with lowercase “c”)

D. ClientService (starting with uppercase “C”)

Correct Answer: *C*

Community vote distribution

C (100%)

Tolo01 5 months ago

Selected Answer: C

C is the best answer

upvoted 2 times

zakupower 9 months, 3 weeks ago

Selected Answer: C

id in this case is taken from the method name, also even when Component Scanning is creating beans it decapitalizes the name of the class. To get a bean with capital first letter you need to specify the name yourself.

upvoted 2 times