
Spring Certified #9



A question lead guide to prepare Spring certification



Spring MVC

Which RESTful HTTP verb is used to modify an existing resource/entity?
(select 2)

- PUT
- PATCH
- POST

PUT & PATCH

POST creates a resource.

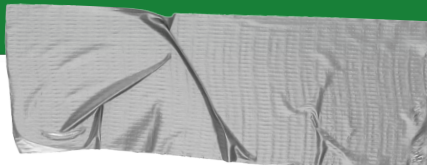
POST can be used to create a new entity, but it should not be used to update an existing entity. The reason is that POST creates a new resource with a new URI, whereas updating an existing entity should modify the existing resource at its existing URI. PUT or PATCH are the HTTP verbs typically used to update an existing entity in REST.

PUT replaces a resource.

The HTTP verb PUT is typically used to update an existing entity/resource in RESTful architecture.

PATCH updates a resource.

The HTTP verb PATCH can also be used in REST in order to update an existing entity. PATCH is often used when you need to update only a subset of the fields of the entity, rather than updating the entire entity as PUT would do. PATCH requests contain a set of instructions on how to modify the resource, usually in the form of a JSON or XML patch document.



Testing

What is the purpose of the `@SpringBootTest` annotation in a Spring Boot application?

- The `@SpringBootTest` annotation is used to indicate that a test is a Spring Boot integration test.
- The `@SpringBootTest` annotation loads the complete Spring application context.
- The `@SpringBootTest` annotation provides full application functionality during testing.
- All of the above.

All of the above.

@interface **SpringBootTest**

Annotation that can be specified on a test class that runs SpringBoot-based tests. Provides the following features over and above the regular *Spring TestContext Framework*:

- Uses [SpringBootTestContextLoader](#) as the default [ContextLoader](#) when no specific [@ContextConfiguration\(loader=...\)](#) is defined.
- Automatically searches for a [@SpringBootConfiguration](#) when nested [@Configuration](#) is not used, and no explicit [classes](#) are specified.
- Allows custom [Environment](#) properties to be defined using the [properties attribute](#).
- Allows application arguments to be defined using the [args attribute](#).
- Provides support for different [webEnvironment](#) modes, including the ability to start a fully running web server listening on a [defined](#) or [random](#) port.
- Registers a [TestRestTemplate](#) and/or [WebTestClient](#) bean for use in web tests that are using a fully running web server.

<https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/test/context/SpringBootTest.html>



Spring AOP

Which AOP advice type is executed only after the advised method, regardless of whether it completes normally or throws an exception?

- **After advice**
- **Before advice**
- **Around advice**
- **Introduction**

After advice

In AOP, the following are the four types of advice:

1. After advice: This advice executes code after the advised method, regardless of whether it throws an exception or completes normally.
2. Before advice: This advice executes code before the advised method is invoked.
3. Around advice: This advice executes code before and after the advised method, allowing you to modify the method's input and output.
4. Introduction: Introductions (known as inter-type declarations in AspectJ) enable an aspect to declare that advised objects implement a given interface, and to provide an implementation of that interface on behalf of those objects.



<https://bit.ly/2v7222>



<https://spring-book.mystrikingly.com>