Spring Certified #4

A question lead guide to prepare Spring certification

Configuration

What are some valid ways to perform dependency injection in Spring? (select 2)

- ➔ **Constructor injection**
- ➔ **Setter injection**
- ➔ **Using the new keyword to create an instance of the dependency**
- ➔ **Using a static method to create an instance of the dependency**

- ➔ **Constructor injection**
- ➔ **Setter injection**

Constructor: inject deps via constructor args (Java `@Bean`/`@Configuration` or XML `<constructor-arg>`). Best for required deps.
Setter: inject via setters (Java config or XML `<property>`). Good for optional deps.

Two others are wrong because both **bypass the Spring container**:

- **`new` keyword:** hard-codes the dependency, so Spring can't manage its lifecycle/scope, can't apply AOP/proxies (`@Transactional`, `@Async`, security), can't inject config, and it's hard to mock in tests.

- **Static factory method (`SomeDep.create()`):** same bypass + adds tight coupling/global-ish behavior; hard to override/replace per profile, no bean scoping, no post-processors, and poor testability.

Spring MVC

What is the default embedded web server used by Spring Boot?

- ➔ **Tomcat**
- ➔ **Jetty**
- ➔ **Undertow**
- ➔ **Spring Boot does not use an embedded web server by default.**

Tomcat

- For servlet stack applications, the `spring-boot-starter-web` includes Tomcat by including `spring-boot-starter-tomcat`, but you can use `spring-boot-starter-jetty` or `spring-boot-starter-undertow` instead.

https://docs.spring.io/spring-boot/how-to/webserver.html#howto.webserver.use-another

Testing

______ is a Spring Boot annotation used for testing MVC controllers in an isolated environment, by auto-configuring only the necessary components for testing the web layer of an application.

- ➔ @WebMvcTest
- ➔ @SpringMvcTest
- ➔ @WebLayerTest
- ➔ @WebContextTest

WebMvcTest

Typically `@WebMvcTest` is used in combination with `@MockBean` or `@Import` to create any collaborators required by your `@Controller` beans.

If you are looking to load your full application configuration and use MockMVC, you should consider `@SpringBootTest` combined with `@AutoConfigureMockMvc` rather than this annotation.

https://docs.spring.io/spring-boot/api/java/org/springframework/boot/test/autoconfigure/web/servlet/WebMvcTest.html https://bit.ly/2v7222