Spring Certified #5

A question lead guide to prepare Spring certification

Spring AOP

What is the meaning of the term "aspect" in Spring AOP?

- ➔ **An aspect in Spring AOP is a module that encapsulates a cross-cutting concern, such as logging or security.**
- ➔ **An aspect in Spring AOP is a class that represents a specific point in the execution of a program.**
- ➔ **An aspect in Spring AOP is a design pattern that helps developers structure their code.**
- ➔ **An aspect in Spring AOP is a way to define data models for database access.**

An aspect in Spring AOP is a module that encapsulates a cross-cutting concern, such as logging or security.

Aspect: a modularization of a concern that cuts across multiple classes. Transaction management is a good example of a crosscutting concern in J2EE applications. In Spring AOP, aspects are implemented using regular classes or regular classes annotated with the @Aspect annotation.

https://docs.spring.io/spring-framework/reference/core/aop/introduction-defn.html

Security

What are some features of Spring Security?

- ➔ **Authentication and Authorization**
- ➔ **Configurability**
- ➔ **Integration**
- ➔ **All of the above**

➔ **All of the above**

Authentication and Authorization: Spring Security provides a comprehensive authentication and authorization framework for securing Spring-based applications. It allows you to authenticate users, authorize access to resources, and manage user sessions.

Configurability: Spring Security is highly configurable and can be customized to meet the specific security requirements of an application. It provides a wide range of authentication mechanisms, including form-based login, HTTP basic authentication, and token-based authentication.

Integration: Spring Security integrates with various other Spring frameworks such as Spring MVC, Spring Data, and Spring Boot. It also supports integration with other security providers like LDAP, OAuth, and SAML.

https://docs.spring.io/spring-security/reference/features/index.html

Spring Core

_____ is a Spring annotation that specifies bean dependencies, ensuring that the dependent beans are initialized first. It can be used to solve circular dependencies in Spring.

- ➔ **@DependsOn**
- ➔ **@Autowired**
- ➔ **@Primary**
- ➔ **@Configuration**

@DependsOn

The **@DependsOn** annotation is a Spring annotation used to express a bean dependency relationship. It is used to indicate that the current bean depends on other bean(s) to be initialized first. By using this annotation, we can ensure that the dependent bean is initialized before the dependent-on bean. The **@DependsOn** annotation can be used at the class level of a Spring bean definition, and it takes the name(s) of the dependent bean(s) as its value. When a bean with this annotation is initialized, Spring first initializes all the beans that it depends on, before initializing the bean with the **@DependsOn** annotation.

- `@Autowired`: used to automatically wire dependencies into a Spring-managed bean
- `@Primary`: used to indicate the primary bean when multiple beans of the same type are present
- `@Configuration`: used to indicate that a class provides bean definitions for the application context

https://bit.ly/2v7222

https://bit.ly/springcertbook