**Trả lời câu hỏi Audit Module 4**

1. Xử lý request trong Spring MVC:

- Spring MVC sử dụng Front Controller Pattern, với DispatcherServlet là trung tâm của mô hình.

- DispatcherServlet tiếp nhận request, xác định controller phù hợp và gọi phương thức xử lý.

- Controller nhận request, xử lý logic nghiệp vụ, trả về ModelAndView.

- ViewResolver sẽ chọn view phù hợp để hiển thị kết quả.

2. Dependency Injection trong Spring:

- Dependency Injection (DI) là một nguyên lý lập trình, cho phép loại bỏ sự phụ thuộc trực tiếp giữa các thành phần trong ứng dụng.

- Trong Spring, các bean được quản lý bởi IoC Container, các bean sẽ được tiêm (inject) vào các thành phần khác thay vì tự tạo ra chúng.

3. Cách thực hiện Dependency Injection:

- Constructor Injection: Tiêm dependency thông qua constructor của bean.

- Setter Injection: Tiêm dependency thông qua phương thức setter của bean.

- Interface Injection: Tiêm dependency thông qua một interface được định nghĩa riêng.

4. Framework và Library:

- Framework là một nền tảng phần mềm cung cấp một cấu trúc và các công cụ để phát triển ứng dụng. Framework định hướng và kiểm soát luồng chương trình.

- Library là một bộ sưu tập các hàm, lớp, module được tái sử dụng. Library cung cấp các chức năng cụ thể mà ứng dụng có thể sử dụng.

5. Spring Framework:

- Spring Framework là một framework mã nguồn mở, giúp phát triển các ứng dụng Java Enterprise.

- Spring cung cấp các module để xây dựng các ứng dụng web, ứng dụng desktop, ứng dụng enterprise.

6. Lợi ích của Spring Framework:

- Giảm sự phức tạp của ứng dụng thông qua Dependency Injection.

- Tách biệt logic nghiệp vụ với hạ tầng kỹ thuật.

- Cung cấp các module và công cụ để xây dựng các ứng dụng phức tạp.

- Tích hợp tốt với các framework khác như Hibernate, JSF, Struts...

7. Nguyên lý Inversion of Control:

- Inversion of Control (IoC) là một nguyên lý lập trình, trong đó việc tạo ra và quản lý các đối tượng không được thực hiện bởi chính các đối tượng đó, mà do một code bên ngoài quản lý.

- Trong IoC, các đối tượng được "đảo ngược" quyền điều khiển, không còn tự tạo ra chính mình mà do một container bên ngoài tạo ra và quản lý.

8. Bean trong Spring:

- Bean là các đối tượng Java được quản lý bởi Spring IoC Container.

- Các bean được định nghĩa trong tệp cấu hình XML hoặc bằng cách sử dụng các Annotation.

9. Bean Scope trong Spring:

- Singleton (mặc định): Chỉ có duy nhất một instance trong toàn bộ ứng dụng.

- Prototype: Mỗi lần truy cập sẽ tạo ra một instance mới.

- Request: Tạo ra một instance mới cho mỗi HTTP request.

- Session: Tạo ra một instance mới cho mỗi HTTP session.

- Application: Tạo ra một instance mới cho toàn bộ ứng dụng web.

- WebSocket: Tạo ra một instance mới cho mỗi WebSocket session.

10. @Autowired trong Spring:

- @Autowired là Annotation cho phép Spring tự động tiêm (inject) các dependency vào các bean.

- Spring sẽ tự động tìm kiếm và tiêm các dependency phù hợp vào các bean khi cần.

11. @Component trong Spring:

- @Component là Annotation được sử dụng để đánh dấu một class là một Spring Bean.

- Các bean được đánh dấu bằng @Component sẽ được Spring IoC Container quản lý.

12. Front Controller trong Spring MVC:

- Front Controller là một thiết kế mẫu trong Spring MVC, trong đó DispatcherServlet đóng vai trò là Front Controller.

- DispatcherServlet là trung tâm của Spring MVC, nhận và xử lý tất cả các request từ client.

13. ModelAndView trong Spring MVC:

- ModelAndView là một interface trong Spring MVC, đại diện cho cả Model và View.

- Controller sẽ trả về một ModelAndView, chứa dữ liệu Model và tên View để hiển thị.

14. ModelMap trong Spring MVC:

- ModelMap là một implementation của Model interface trong Spring MVC.

- ModelMap được sử dụng để chứa các dữ liệu cần thiết để hiển thị trên View.

15. ViewResolver trong Spring MVC:

- ViewResolver là một interface trong Spring MVC, dùng để xác định View cần hiển thị dựa trên tên View.

- ViewResolver sẽ tìm kiếm và trả về View phù hợp dựa trên tên View được trả về từ Controller.

17. POST và PUT:

- POST thường được sử dụng để tạo mới một tài nguyên trên server.

- PUT thường được sử dụng để cập nhật một tài nguyên đã tồn tại trên server.

18. @RequestMapping trong Spring MVC:

- @RequestMapping là Annotation dùng để ánh xạ một URL request đến một phương thức xử lý trong Controller.

- @RequestMapping cho phép chỉ định phương thức HTTP, URL, parameters, headers, etc.

19. Data Binding trong Spring:

- Data Binding là quá trình chuyển đổi dữ liệu từ request (như form data) sang các đối tượng Java.

- Spring sử dụng các Converter và Formatter để thực hiện Data Binding.

20. consumes trong Request Mapping:

- Thuộc tính consumes trong @RequestMapping chỉ định các media type mà request body phải tuân thủ.

- Nó giúp Spring MVC xác định được request body có dữ liệu ở định dạng nào.

21. Thymeleaf:

- Thymeleaf là một template engine dành cho Java, được sử dụng để tạo ra các trang web động.

- Thymeleaf cung cấp các tính năng như hiển thị dữ liệu, xử lý điều kiện, lặp, etc.

22. Sử dụng lặp và điều kiện trong Thymeleaf:

- Lặp: Sử dụng th:each để lặp qua các phần tử của một danh sách.

- Điều kiện: Sử dụng th:if, th:unless, th:switch, th:case để kiểm tra và hiển thị dữ liệu dựa trên điều kiện.

23. Formatter và Converter:

- Formatter là interface trong Spring dùng để format dữ liệu khi binding từ/sang String.

- Converter là interface trong Spring dùng để chuyển đổi giữa các kiểu dữ liệu khác nhau.

24. ORM (Object-Relational Mapping):

- ORM là một kỹ thuật lập trình cho phép ánh xạ giữa các đối tượng trong ứng dụng Java và bảng trong cơ sở dữ liệu.

- ORM giúp lập trình viên có thể tương tác với cơ sở dữ liệu thông qua các đối tượng Java thay vì câu lệnh SQL.

25. JPA và Spring Data JPA:

- JPA (Java Persistence API) là một API chuẩn cho ORM trong Java.

- Spring Data JPA là một module của Spring Framework, cung cấp các tiện ích để làm việc với JPA dễ dàng hơn.

26. Hibernate:

- Hibernate là một framework ORM (Object-Relational Mapping) mã nguồn mở cho Java.

- Hibernate giúp map các đối tượng Java với các bảng trong cơ sở dữ liệu quan hệ.

27. Một số annotation của Hibernate:

- @Entity: Đánh dấu class là một Entity.

- @Id: Đánh dấu trường là khóa chính.

- @Column: Ánh xạ trường với cột trong bảng.

- @OneToMany, @ManyToOne: Đánh dấu quan hệ giữa các Entity.

28. Entity trong Hibernate:

- Entity là các class đại diện cho các bảng trong cơ sở dữ liệu.

- Các Entity được ánh xạ với các bảng bằng các Annotation hoặc XML mapping.

29. Persistence Context và Entity Manager:

- Persistence Context là môi trường quản lý vòng đời của các Entity.

- Entity Manager là interface cung cấp các phương thức để tương tác với Persistence Context và cơ sở dữ liệu.

30. Truy vấn tĩnh và truy vấn động:

- Truy vấn tĩnh là các truy vấn được định nghĩa sẵn, có cú pháp cố định.

- Truy vấn động là các truy vấn có thể thay đổi điều kiện tại thời điểm thực thi.

31. Trạng thái của Entity:

- Transient: Entity chưa được lưu vào cơ sở dữ liệu.

- Persistent: Entity đã được lưu vào cơ sở dữ liệu và được quản lý bởi Persistence Context.

- Detached: Entity không còn được quản lý bởi Persistence Context.

- Removed: Entity đã được đánh dấu để xóa khỏi cơ sở dữ liệu.

32. Các loại mapping trong Hibernate:

- One-to-One

- One-to-Many

- Many-to-One

- Many-to-Many

33. SessionFactory trong Hibernate:

- SessionFactory là single-threaded, thread-safe object được sử dụng để tạo ra các Session.

- SessionFactory được khởi tạo dựa trên file cấu hình Hibernate.

34. Validation dữ liệu:

- Validation là quá trình kiểm tra tính hợp lệ của dữ liệu trước khi sử dụng hoặc lưu trữ.

- Validation giúp đảm bảo dữ liệu nhập vào đúng định dạng và không vi phạm các ràng buộc nghiệp vụ.

35. Triển khai validation trong Spring:

- Sử dụng Annotation như @NotNull, @Size, @Email, etc. để định nghĩa các ràng buộc.

- Sử dụng @Valid để kích hoạt validation khi binding dữ liệu từ request.

- Xử lý lỗi validation bằng BindingResult.

36. BindingResult trong Spring:

- BindingResult là interface chứa thông tin về các lỗi validation xảy ra trong quá trình binding dữ liệu.

- BindingResult được sử dụng để lấy thông tin lỗi và hiển thị chúng trên giao diện.

37. AOP (Aspect-Oriented Programming):

- AOP là một phương pháp lập trình cho phép tách biệt các quan ngại (concerns) liên quan đến hệ thống ra khỏi mã nguồn chính.

- AOP bao gồm các khái niệm như Aspect, Advice, Joinpoint, Pointcut.

38. Các loại Advice trong AOP:

- Before: Thực thi trước khi gọi phương thức.

- After: Thực thi sau khi gọi phương thức.

- Around: Thực thi xung quanh việc gọi phương thức.

- AfterReturning: Thực thi sau khi phương thức trả về.

- AfterThrowing: Thực thi khi phương thức ném ngoại lệ.

39. JoinPoint trong AOP:

- JoinPoint là điểm trong chương trình nơi mà Aspect có thể can thiệp.

- JoinPoint cung cấp thông tin về phương thức đang được gọi, đối số, etc.

40. Triển khai AOP trong dự án:

- Sử dụng Annotation như @Aspect, @Pointcut, @Before, @After, @Around, etc.

- Hoặc cấu hình AOP thông qua XML.

41. Xử lý ngoại lệ trong Spring Web MVC:

- Sử dụng Annotation @ExceptionHandler trong Controller để xử lý ngoại lệ.

- Có thể sử dụng HandlerExceptionResolver để xử lý ngoại lệ toàn cục.

42. Cookie:

- Cookie là dữ liệu ngắn được lưu trữ trong trình duyệt của người dùng.

- Cookie được sử dụng để lưu trữ thông tin về phiên làm việc, thông tin cá nhân, cài đặt người dùng, etc.

43. Session và Cookie:

- Session lưu trữ thông tin liên quan đến người dùng trên server.

- Cookie lưu trữ thông tin liên quan đến người dùng trên client.

44. Triển khai Session trong dự án:

- Sử dụng HttpSession để lưu trữ và truy xuất thông tin session.

- Cấu hình Session Timeout, Session Tracking Mode, etc.

45. Web Service:

- Web Service là một ứng dụng phần mềm được thiết kế để hỗ trợ tương tác máy-máy qua mạng.

- Ví dụ: API cung cấp thông tin thời tiết, tỷ giá hối đoái, etc.

46. Phân biệt Web Service và Web truyền thông

- Web Service và Web truyền thông khác nhau về mục đích: Web Service để trao đổi dữ liệu giữa các ứng dụng, Web truyền thông để tương tác với người dùng.

- Web Service không có giao diện người dùng, chỉ cung cấp API, còn Web truyền thông cung cấp giao diện HTML, CSS, JS.

- Web Service sử dụng các giao thức như SOAP, REST, Web truyền thông sử dụng HTTP.

- Web Service sử dụng định dạng chuẩn như XML, JSON, Web truyền thông sử dụng các định dạng như HTML, CSS, JS.

-Web Service được thiết kế để tái sử dụng, Web truyền thông thường ít được tái sử dụng.

47. SOAP và RESTful:

- SOAP (Simple Object Access Protocol) là một giao thức dựa trên XML để trao đổi thông tin giữa các ứng dụng.

- RESTful (Representational State Transfer) là một kiến trúc phần mềm dựa trên HTTP để trao đổi thông tin giữa các ứng dụng.

48. jQuery và Ajax:

- jQuery là một thư viện JavaScript giúp việc thao tác DOM, xử lý sự kiện, và gọi Ajax dễ dàng hơn.

- Ajax (Asynchronous JavaScript and XML) là một kỹ thuật sử dụng JavaScript để tạo ra các ứng dụng web có tính tương tác cao.

49. Triển khai Ajax trong dự án:

- Sử dụng jQuery.ajax() hoặc $.get(), $.post() để thực hiện các cuộc gọi Ajax.

- Xử lý kết quả trả về và cập nhật giao diện người dùng.

50. @RequestHeader và @ResponseHeader:

- @RequestHeader dùng để lấy giá trị từ header của request.

- @ResponseHeader dùng để thêm header vào response.

51. I18N và L10N:

- I18N (Internationalization) là quá trình thiết kế và phát triển sản phẩm để có thể được sử dụng ở nhiều ngôn ngữ và văn hóa khác nhau.

- L10N (Localization) là quá trình thích ứng sản phẩm để phù hợp với một ngôn ngữ và văn hóa cụ thể.

52. Triển khai I18N và L10N trong Spring:

- Sử dụng ResourceBundleMessageSource để quản lý các tài nguyên đa ngôn ngữ.

- Sử dụng Locale để xác định ngôn ngữ hiển thị.

- Sử dụng @MessageSource và #{} để lấy và hiển thị các thông báo đa ngôn ngữ.

53. Interceptor trong Spring MVC:

- Interceptor là một cơ chế cho phép can thiệp vào quá trình xử lý request trước khi đến Controller.

- Interceptor thường được sử dụng để xác thực, phân quyền, ghi log, etc.

54. Bảo mật trong Spring:

- Spring Security cung cấp một framework để xây dựng các ứng dụng an toàn.

- Spring Security hỗ trợ các cơ chế xác thực và phân quyền linh hoạt.

55. Authentication và Authorization:

- Authentication là quá trình xác định danh tính của người dùng.

- Authorization là quá trình kiểm tra quyền hạn của người dùng để thực hiện một hành động cụ thể.

56. Các cơ chế xác thực:

- Basic Authentication

- Digest Authentication

- Form-based Authentication

- OAuth 2.0

- OpenID Connect

57. CSRF (Cross-Site Request Forgery):

- CSRF là một lỗ hổng bảo mật cho phép kẻ tấn công thực hiện các hành động không mong muốn thay mặt người dùng.

- Spring Security cung cấp bảo vệ CSRF mặc định.

58. CORS (Cross-Origin Resource Sharing):

- CORS là một cơ chế cho phép các tài nguyên được truy cập từ một nguồn khác với nguồn trang web.

- Spring MVC cung cấp hỗ trợ CORS thông qua @CrossOrigin annotation.

59. Spring Boot:

- Spring Boot là một module của Spring Framework, giúp tạo ra các ứng dụng Spring một cách nhanh chóng.

- Spring Boot cung cấp các tính năng sẵn như Embedded Servlet Container, Auto-configuration, Starter Dependencies.

60. Sự khác nhau giữa Spring Boot và Spring MVC:

- Spring MVC là một framework web, trong khi Spring Boot là một framework để tạo ra các ứng dụng Spring một cách dễ dàng hơn.

- Spring Boot cung cấp các tính năng sẵn như Embedded Servlet Container, Auto-configuration, Starter Dependencies.

61. Cấu hình Spring Boot:

- Sử dụng @ComponentScan để quét các bean ở nhiều package khác nhau.

- Hoặc khai báo các package cần quét trong application.properties/application.yaml.

62. Spring Security:

- Spring Security là một framework bảo mật mạnh mẽ cho các ứng dụng Spring.

- @EnableWebSecurity kích hoạt Spring Security và cho phép cấu hình bảo mật web.

63. Spring Boot JPA:

- Spring Boot JPA cung cấp hỗ trợ tích hợp giữa Spring Boot và JPA.

- Cài đặt Spring Boot JPA bằng cách thêm spring-boot-starter-data-jpa vào dependencies.

64. Sử dụng JpaRepository:

- Để sử dụng JpaRepository, cần khai báo một interface kế thừa JpaRepository.

- Trong interface, chỉ cần định nghĩa các method, Spring Data JPA sẽ tự động triển khai.

65. @ConfigurationProperties:

- @ConfigurationProperties cho phép đọc các cấu hình từ file application.properties/application.yaml vào các class Java.

- Sử dụng prefixes để nhóm các cấu hình liên quan với nhau.