|  |  |
| --- | --- |
| 1. Framework là gì? Phân biệt Framework và Library? |  **Framework**: là một bộ khung, tập hợp các thư viện, công cụ, quy tắc... hỗ trợ lập trình viên phát triển ứng dụng dễ dàng hơn, có cấu trúc rõ ràng. Framework quy định cách mà ứng dụng phải được tổ chức.   **Library**: là tập hợp các hàm, class... mà lập trình viên có thể gọi để thực hiện một chức năng nào đó. |
| 2. Luồng đi của 1 request trong Spring MVC | Luồng xử lý request:   1. **Client** gửi request → DispatcherServlet. 2. DispatcherServlet tìm **HandlerMapping** để xác định Controller xử lý request. 3. **Controller** nhận request, xử lý logic và trả về tên View + dữ liệu (Model). 4. DispatcherServlet nhờ **ViewResolver** tìm ra View tương ứng. 5. View được render → trả HTML về cho **Client**. |
| 3. Phân biệt Spring MVC và MVC |  **MVC (Model-View-Controller)**: mô hình kiến trúc chung, tách ứng dụng thành:   * Model (dữ liệu, logic), * View (giao diện hiển thị), * Controller (xử lý request).    **Spring MVC**: là một framework triển khai mô hình MVC dựa trên Spring, cung cấp các thành phần như:   * DispatcherServlet, * Controller, * ViewResolver, * HandlerMapping, * Hỗ trợ DI/IoC của Spring. |
| 4. ViewResolver là gì? |  **ViewResolver** là thành phần trong Spring MVC giúp ánh xạ tên View (chuỗi mà Controller trả về) → một file thực tế (JSP, Thymeleaf, v.v.).   Ví dụ:   * Controller return "home" * ViewResolver tìm và render /WEB-INF/views/home.jsp. |
| 5. Cách chuyển và nhận dữ liệu ở View và Controller trong Spring MVC | **Controller → View**: sử dụng Model, ModelMap, ModelAndView.  **View → Controller**: dùng @RequestParam, @ModelAttribute, @RequestBody. |
| 6. Bean là gì? Các cách định nghĩa bean? | **Bean**: là đối tượng được Spring Container quản lý (tạo, gán dependency, hủy).   **XML configuration**  <bean id="student" class="com.example.Student"/>   **Annotation (@Component, @Service, @Repository, @Controller)**   **Java-based configuration (@Bean)**  @Configuration  public class AppConfig {  @Bean  public Student student() {  return new Student();  }  } |
| 7. IoC là gì? DI là gì? Các cách thực hiện DI? |  **IoC (Inversion of Control)**: đảo ngược quyền kiểm soát việc khởi tạo và quản lý đối tượng. Thay vì dev tự new, Spring Container lo việc đó.   **DI (Dependency Injection)**: kỹ thuật để thực hiện IoC – Spring sẽ "tiêm" dependency vào đối tượng khi cần. |
| 8. Spring Container là gì? | * **Spring Container**: là bộ máy cốt lõi của Spring, chịu trách nhiệm:   + Tạo và quản lý vòng đời của bean.   + Thực hiện Dependency Injection.   + Cung cấp các dịch vụ hỗ trợ (AOP, event, resource loading...).   👉 Có 2 loại container chính:   1. **BeanFactory**: đơn giản, chỉ khởi tạo bean khi cần (lazy loading). 2. **ApplicationContext**: mạnh mẽ hơn, hỗ trợ nhiều tính năng (event, i18n, AOP), thường dùng nhất. |