|  | **Câu hỏi** | **Trả lời ngắn gọn** | **Ghi chú / Ví dụ** |
| --- | --- | --- | --- |
|  | **Framework là gì? Phân biệt Framework và Library?** | **Framework**: bộ khung có quy ước, vòng đời và “luật chơi”; ứng dụng của bạn **cắm vào** và đi theo. **Library**: bộ hàm/tính năng bạn **gọi khi cần**. | Khác biệt cốt lõi: **IoC/“who calls who”** – với framework *nó gọi code của bạn*; với library *bạn gọi code của nó*. VD: Spring (framework) vs Apache Commons/Guava (library). |
|  | **Luồng đi của 1 request trong Spring MVC?** | Client → **Filter** → **DispatcherServlet** → **HandlerMapping** tìm controller → (optional **Interceptor preHandle**) → **Controller** gọi **Service/Repository** → trả **Model & view name** hoặc **body** → **ViewResolver** tìm view (Thymeleaf/JSP) **hoặc** **HttpMessageConverters** (REST) → (Interceptor post/after) → Response. | Thêm: **ExceptionResolver** bắt lỗi, **RedirectAttributes** cho redirect, **@ResponseBody/@RestController** đi qua converters (JSON). |
|  | **Phân biệt Spring MVC và MVC?** | **MVC**: mẫu kiến trúc (Model–View–Controller) trừu tượng. **Spring MVC**: **framework** hiện thực MVC trong Spring, cung cấp DispatcherServlet, annotations, binding, validation, view resolvers… | Tức là Spring MVC **áp dụng** MVC + rất nhiều tiện ích sẵn có. |
|  | **ViewResolver là gì?** | Thành phần map **tên view logic** (vd "users/detail") sang **template thực** (vd /templates/users/detail.html). | Các loại: ThymeleafViewResolver, InternalResourceViewResolver (JSP). Có **prefix/suffix** và **order** khi nhiều resolver. |
|  | **Chuyển/nhận dữ liệu giữa View & Controller?** | **Controller → View**: Model/ModelMap, ModelAndView, @ModelAttribute, RedirectAttributes.addFlashAttribute(). **View → Controller**: @RequestParam (query/form), @PathVariable (URL), @ModelAttribute (bind form object), @RequestBody (JSON), upload file MultipartFile. | Ví dụ: model.addAttribute("user", u); – form POST bind vào @ModelAttribute UserDto dto. |
|  | **Bean là gì? Các cách định nghĩa bean?** | **Bean**: đối tượng do **Spring IoC container** tạo/quản lý vòng đời & dependencies. | Khai báo: 1) **Stereotype** @Component/@Service/@Repository/@Controller + **component scan**; 2) @Configuration + **@Bean** method; 3) (cũ) XML <bean>. Tuỳ chọn: @Primary, @Qualifier, cấu hình scope (singleton, prototype, request, session…). |
|  | **IoC là gì? DI là gì? Triển khai DI?** | **IoC** (Inversion of Control): chuyển quyền tạo & quản lý đối tượng sang container. **DI** (Dependency Injection): container **bơm phụ thuộc** vào bean. | Cách DI: **Constructor injection** (khuyến nghị, bất biến/dễ test), **Setter injection**, **Field injection** (ít khuyến nghị). Dùng @Autowired (hoặc constructor 1 tham số không cần ghi), @Qualifier, @Value cho cấu hình. |
|  | **Spring Container là gì?** | Thành phần cốt lõi thực thi IoC/DI: **tạo, wire, khởi tạo, quản lý vòng đời** bean, xử lý **AOP**, sự kiện, resource loading. | Hiện thực chính: **ApplicationContext** (phổ biến) và **BeanFactory** (cơ bản). AnnotationConfigApplicationContext, SpringApplication.run(...) sẽ khởi tạo container. |