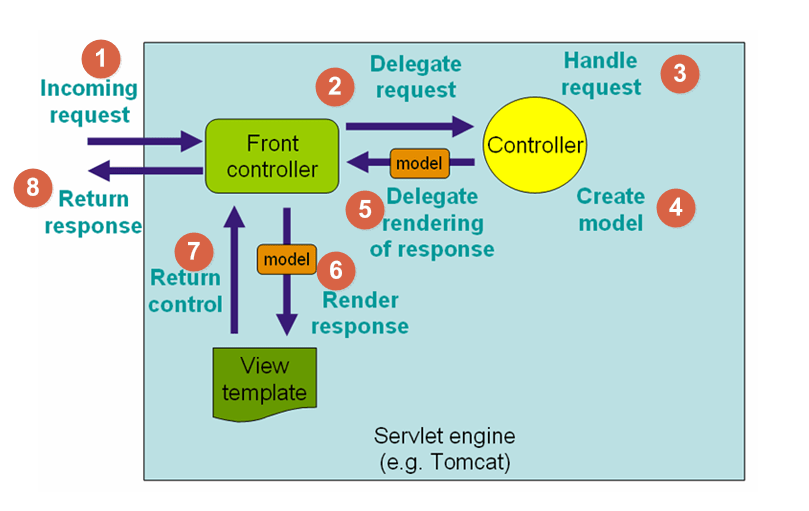
**Audit module 4**

1. **Vẽ và trình bày cách Spring MVC xử lý request?**



1. Front-controller sẽ là nơi đầu tiên nhận tất cả request người dùng gửi đến
2. Front-controller sẽ sử dụng Handle mapping để biết controller nào sẽ xử lý request đó
3. Controller nhận request, gọi các class service xử lý liên quan
4. Sau khi xử lý xong, Controller nhận được model từ tầng Service
5. Controller gửi model vừa nhận được tới Front-controller
6. Front-controller sẽ sử dụng các mẫu view, sử dụng View Resolver và truyền model vào nó
7. View template, view page, model sẽ được gửi lại Front-controller
8. Front-controller gửi một page view để trình duyệt hiển thị ra cho người dùng
9. **Trình bày cơ chế Dependency Injection trong Spring**

DI là cơ chế để loại bỏ sự phụ thuộc trong chương trình, giúp cho việc quản lý và kiểm thử phần mềm dễ dàng hơn. DI làm cho mã chương trình ít bị phụ thuộc vào nhau hơn

DI là một dạng thực hiện của IoC, bằng cách tiêm (inject) module vào một module khác cần nó. Ví dụ module cấp cao Car cần có module Engine, thì dependency injection sẽ thực hiện theo các bước như sau:

* Tìm và tạo module tương ứng với Engine (class nào có implements Engine interface)
* Tạo tiếp module Car, do Car phụ thuộc vào Engine nên IoC tìm và lấy ra đối tượng Engine đã tạo trước đó và inject vào bên trong Car.
* Xong

Mọi module trong IoC đều gọi là dependency, mặc dù có những module không bị phụ thuộc bởi module nào khác. Khi chương trình chạy, IoC sẽ quét tất cả class đánh dấu dependency, tạo một đối tượng duy nhất (singleton), và bỏ vào cái túi gọi là IoC container, lúc nào cần thì lấy ra sử dụng. Do đó, các module đảm bảo được IoC tạo ra duy nhất một object, giúp tiết kiệm bộ nhớ và quản lý cũng dễ hơn.

Nếu khi tạo module nào đó, mà module đó cần một module khác phụ thuộc, thì IoC sẽ tìm trong IoC container xem có không, nếu có thì inject vào, nếu chưa thì tạo mới, bỏ vào container và inject vào. Việc inject tự động các dependency (module) như thế được gọi là **Dependency injection**.

1. **Có bao nhiêu cách để thực hiện Dependency Injection? Trình bày?**

Có 3 loại cách:

* Constructor-based injection
* Setter-based injection
* Field injection

|  |  |
| --- | --- |
| **Constructor-based injection** | **Setter-based injection** |
| Inject những module bắt buộc | Inject module tùy chọn |
| Có thể inject nhiều dependency cùng lúc | Chỉ inject mỗi dependency mỗi setter |

1. **Framework là gì? Framework khác Library chỗ nào?**

* Framework là các đoạn code được viết sẵn, cấu thành nên một bộ khung và thư viện lập trình được đóng gói. Chúng cung cấp các tính năng có sẵn như mô hình, API và các yếu tố khác để tối giản cho việc lập trình web.
* Library: là tập hợp các chức năng, các lớp được viết sẵn để có thể tái sử dụng. Mỗi function, class phục vụ cho một công việc cụ thể nào đó.

|  |  |
| --- | --- |
| **Framework** | **Library** |
| * Cung cấp các tính năng có sẵn để có thể tái sử dụng | |
| * LTV phải tuân theo cấu trúc, quy tắc của framewok | * LTV sử dụng các tính năng của library để thêm vào dự án của mình |

1. **Spring framework là gì?**

* Spring framework là một mã nguồn mở dùng để phát triển các ứng dụng Java, giúp LTV phát triển ứng dụng có hiệu suất cao, dễ kiểm thử và tái sử dụng.

Các thành phần của Spring:

* Spring core: core, bean, context, SpEL
* Testing: cung cấp các module để test
* Spring Data Acess: module cung cấp cách tương tác DL
* Web Servlet: cung cấp mô hình ứng dụng web
* Web Reactive
* Intergration
* Language: Java, Kotlin, Groovy

1. **Lợi ích của Spring framework?**

* Tăng tốc độ phát triển
* Giúp LTV tập trung xử lý nghiệp vụ
* Cung cấp môi trường làm việc tiêu chuẩn
* Cộng đồng lớn mạnh

1. **Nguyên lý đảo ngược điều khiển(Inversion of Control) là gì?**

* IoC: đảo ngược điều khiển, giúp thay đổi luồng điều khiển của chương trình một cách linh hoạt. IoC container là thành phần thực hiện IoC
* Trong Spring, IoC container sẽ tạo các đối tượng, lắp ráp chúng lại với nhau, cấu hình các đối tượng và quản lý vòng đời của chúng
* Spring container sử dụng DI để quản lý các thành phần, đối tượng để tạo nên 1 ứng dụng. Các thành phần, đối tượng này được gọi là Spring Bean.
* Bean Factory và ApplicationContext đều là các interface để thực hiện Spring Container. ApplicationContext có nhiều chức năng hơn: AOP, xử lý message, context cho web application.

1. **Bean là gì?**

* Bean là 1 đối tượng Java được tạo ra và quản lý bởi Spring container.

1. **Trong Spring có bao nhiêu Bean Scope?**

Có 6 bean scope trong Spring:

* Singleton scope(mặc định): IoC container sẽ khởi tạo duy nhất 1 instance cho bean đó và nó được sử dụng trong tất cả các yêu cầu đến bean này.
* Prototype scope: sẽ trả về 1 instance mới khi có yêu cầu đến IoC container
* Request scope: khởi tạo 1 bean instance cho một Http request
* Session scope: khởi tạo 1 bean instance cho 1 Http session
* Application scope: khởi tạo 1 bean instance cho 1 vòng đời của ServletContext
* Websocket scope: khởi tạo cho 1 websocket session

1. **@Autowire là gì?**

* @Autowire cho phép Spring tự động tìm kiếm và inject các bean tương ứng mà chúng ta đã khai báo trong class.

Có 3 cách để sử dụng autowire:

* Autowire trực tiếp trên thuộc tính
* Autowire setter
* Autowire constructor

1. **@Component nghĩa là gì?**

* @Component là một annotation đánh dấu trên 1 class để Spring biết nó là 1 Bean. Khi chạy Spring boot gặp 1 class đánh dấu là @Component thì nó sẽ tạo ra 1 instance để đưa vào ApplicationContext quản lý.

1. **Trình bày ý nghĩa của Controller?**

* Controller là 1 thành phần trong mô hình MVC, controller nhận request từ người dùng, xử lý request, xây dựng dữ liệu cho view(model) và chọn view để trả về kết quả cho người dùng

1. **Trình bày ý nghĩa của ModelAndView Interface?**

* Là sự kết hợp giữa Model và View, giúp chúng ta có thể gửi model và view chỉ trong 1 dòng lệnh

1. **ModelMap interface là gì?**

* Là đối tượng implement của Map interface, chúng ta có thể dùng put để đưa dữ liệu vào, ngoài ra nó còn hỗ trợ một số phương thức khác để đưa dữ liệu vào.

1. **Trình bày ý nghĩa của ViewResolver interface?**

* Là các đối tượng chịu trách nhiệm lấy các đối tượng View cho một hành động và vị trí cụ thể. Thông thường, Controller yêu cầu ViewResolvers chuyển tiếp đến một view với một tên cụ thể(một chuỗi được trả về phương thức). Sau đó tất cả ViewResolvers trong ứng dụng thực thi theo thứ tự cho đến khi một trong số đó phân giải view trong trường hợp một đối tượng View trả về và điều khiển được truyền tới nó cho việc render HTML.

1. **Phân biệt POST và PUT thường để làm gì?**

|  |  |
| --- | --- |
| **POST** | **PUT** |
| * Unsafe: có thể thay đổi trạng thái của server, nó có thể chỉnh sửa, tạo mới, xóa thông tin | |
| * POST: gửi thông tin đến server thông qua form * POST: ko idempotent, tức là khi thực hiện cùng 1 nội dung giống nhau 2 lần thì sẽ bị lỗi | * PUT: ghi đè tất cả thông tin của đối tượng gửi lên * PUT là imdempotent |

1. **@RequestMapping để làm gì?**

* @RequestMapping để map tới class hoặc method xử lý request đó.

1. **Trình bày cơ chế DataBinding trong Spring?**

* DataBingding là cơ chế liên kết đối tượng đầu vào(hoặc đầu ra) với các đối tượng model. Sử dụng Databinding các form đều được liên kết với các đối tượng phía sau dữ liệu
* Databinding hỗ trợ chuyển đổi dữ liệu(data conversion) và validate dữ liệu