1.MVC chia thành bao nhiêu tầng? Làm thế nào để biết 1 class là thuộc tầng nào? Bean dùng để làm gì?

**MVC** là viết tắt của **Model-View-Controller**, một mô hình kiến trúc phần mềm được sử dụng để xây dựng các ứng dụng web. Trong mô hình này, ứng dụng được chia thành ba tầng chính:

* **Model**: đại diện cho dữ liệu của ứng dụng và cung cấp các phương thức để truy xuất và thao tác với dữ liệu này.
* **View**: đại diện cho giao diện người dùng của ứng dụng. Nó hiển thị dữ liệu từ model và cung cấp các phương thức để tương tác với người dùng.
* **Controller**: làm trung gian giữa model và view. Nó xử lý các sự kiện và yêu cầu từ người dùng và cập nhật model và view tương ứng.

Một class thuộc tầng nào phụ thuộc vào chức năng của class đó trong ứng dụng. Ví dụ, một class chịu trách nhiệm truy xuất dữ liệu từ cơ sở dữ liệu sẽ thuộc tầng model, trong khi một class xử lý các yêu cầu từ người dùng sẽ thuộc tầng controller.

**Bean** là một đối tượng được quản lý bởi Spring Framework. Nó được tạo ra và quản lý bởi Spring IoC container và được sử dụng để thực hiện các tác vụ như cấu hình ứng dụng, truy xuất dữ liệu và quản lý các thành phần của ứng dụng. Bean thường được sử dụng để phân tách logic ứng dụng và giúp dễ dàng quản lý và bảo trì ứng dụng.

2. Luồng đi của MVC, luồng đi từ client => Front Controller

+ Luồng thực thi:

🡪 Người dùng điền tên **website** cần truy cập trên **brower** 🡪 nhấn **Enter**

🡪 gửi **request** lên **server** (nơi triển khai ứng dụng **Spring**)

🡪 **DispatcherServlet (Front Controller)** tiếp nhận **request**

🡪 chuyển tới **controller** tương ứng(thông qua các cơ chế **mapping –** ánh xạ đường dẫn)

🡪 **controller** nhận yêu cầu và gọi phương thức thích hợp (**GET** / **POST**)

🡪 **controller** tiến hành **render** dữ liệu và trả về tên **view** cho **DispatcherServlet**.

🡪 **DispatcherServlet** sẽ nhận trợ giúp từ **ViewResolver** để xác định **view** cho các yêu cầu.

🡪 **DispatcherServlet** chuyển dữ liệu tới **view** đó 🡪 chuyển tới trình duyệt.

3. Khác nhau giữa Sping MVC và Sping Boot?

**SpringBoot**

1. Là một **module** của **Spring** để đóng gói ứng dụng dựa trên **Spring** với các giá trị mặc định hợp lý.
2. Cung cấp các cấu hình mặc định để xây dựng khung công tác được cung cấp bởi **Spring**.
3. Không cần xây dựng cấu hình theo cách thủ công.
4. Tránh mã viết sẵn và kết hợp các phần phụ thuộc lại với nhau trong một đơn vị duy nhất.
5. Làm giảm thời gian phát triển và tăng năng suất.

**SpringMVC**

1. **Spring MVC** là **model-view-controller** web framework trong khuôn khổ **Spring framework**.
2. Cung cấp các tính năng sẵn sàng sử dụng để xây dựng một ứng dụng **web**.
3. Yêu cầu cấu hình xây dựng theo cách thủ công.
4. Nó chỉ định từng phụ thuộc riêng biệt.
5. Cần nhiều thời gian hơn để phát triển.

4.ORM là gì?

- **ORM** (**Object-Relational Mapping**) – là kỹ thuật liên kết giữa đối tượng trong lập trình với đối tượng trong **CSDL** (là quá trình biến đổi giữa 1 **record** trong **DB** và 1 **object** trong ngôn ngữ lập trình **OOP**).

+ **ORM** ra đời để khắc phục nhược điểm của **JDBC**, như:

- Cấu hình thủ công (cần phải chuẩn bị **DB** trước).

- Có kích thước nền tảng về **SQL**.

- Phải biết kiểu dữ liệu đầu ra / đầu vào của từng **column**.

+ **ORM** có rất nhiều triển khai (Hibernate, Eclipselink, Ibatis,…), trong đó **Hibernate** đc xem là **ORM Framework** nổi tiếng nhất và sử dụng rộng rãi trong **Java**.

5. Chức năng của Entity, Repository, Service, Controller?

|  |
| --- |
| **1/Entity** là đối tượng đại diện cho dữ liệu trong ứng dụng |
| Mỗi Entity sẽ được mapping với một table trong CSDL |
| Mỗi Entity cần tuân thủ: |
| + Được gắn annotation @Entity |
| + Có constructor public rỗng |
| + Không Được khai báo final |
| + Thuộc tính của đối tượng Có access modifier là private, protected hoặc ở mức package private |

6. (Trending) Giải thích DI trong Java? Khi dùng @Autowired thì sẽ luôn tạo ra đối tượng mới hay như thế nào?

1. **DI** (**Dependency Injection**) – là **design pattern** cho phép xóa bỏ sự phụ thuộc, là một cơ chế để triển khai **IoC**, trong đó thao tác tiêm (**inject**) vào các đối tượng khác được thực hiện bởi **container** hoặc **framework**.

(Giống với Nguyên tắc **D** trong **S.O.L.I.D** là **Dependency inversion principle** - **Nguyên lý đảo ngược phụ thuộc**: các **module** cấp cao không nên phụ thuộc vào các **module** cấp thấp, cả 2 nên phụ thuộc vào **abstraction**).

=> Giảm sự phụ thuộc giữa các **module**, **object** giúp cho việc mở rộng **code** sau này được dễ dàng.

=> Giúp cho việc quản lý và kiểm thử ứng dụng dễ dàng hơn.

=> Giúp chúng ta tập trung vào việc viết **logic** nghiệp vụ của ứng dụng, còn việc tạo và quản lý các đối tượng đc giao cho **framework** lo.

🡺 **Có 3 cách triển khai:**

+ **Constructor** - Dùng **inject** (tiêm) các **module** bắt buộc. Các **module** được tiêm nằm trong **constructor**, và được gán lần lượt vào các **field**.

+ **Setter** method - Dùng **inject** các **module** tùy chọn. Mỗi **module** sẽ được **inject** thông qua **setter**, nằm ở tham số và cũng gán cho **field** nào đó.

+ **Interface** (field) - Các **client** phải triển khai một **Interface** mà có một **setter** method dành cho việc nhận **dependency**.

1. Trong Spring Framework, khi sử dụng @Autowired để tiêm phụ thuộc (dependency injection), đối tượng được tạo ra chỉ một lần và sử dụng lại trong các lần gọi tiếp theo của bean đó.

7. (Trending) RequestMapping hiểu như thế nào? RequestMapping dùng cho method vậy thì có dùng được cho Class hay không?

* @RequestMapping là một annotation được sử dụng để ánh xạ các HTTP request đến các method trong controller. Nó giúp xác định các method nào sẽ được gọi để xử lý request dựa trên đường dẫn URL và phương thức HTTP.
* Các tham số của @RequestMapping bao gồm:
* value: Đường dẫn URL mà method sẽ xử lý.
* method: Phương thức HTTP được yêu cầu để xử lý request.
* params: Các tham số request yêu cầu để method được gọi.
* headers: Các header request yêu cầu để method được gọi.
* @RequestMapping có thể được sử dụng cả cho class và method. Nếu bạn đặt annotation @RequestMapping trên class, đường dẫn URL được chỉ định trong annotation đó sẽ được sử dụng làm tiền tố cho các đường dẫn URL trong các phương thức xử lý yêu cầu HTTP.

8. Spring email là gì?

1. **Spring Email** là một module trong Spring Framework cung cấp các tính năng để gửi và nhận email trong ứng dụng Java của bạn.
2. Với Spring Email, bạn có thể gửi email bằng các giao thức SMTP, SMTPS hoặc TLS và đính kèm các file như hình ảnh, tài liệu, văn bản... Bạn cũng có thể tùy chỉnh nội dung của email bằng HTML hoặc Plain text.
3. Các thành phần chính của Spring Email bao gồm:

* **MailSender**: interface cho phép gửi email.
* **JavaMailSenderImpl**: lớp cung cấp một implement của MailSender cho giao thức SMTP và SMTPS.
* **MimeMessageHelper**: lớp cung cấp một số phương thức tiện ích cho việc tạo và cấu hình **MimeMessage** (định dạng email).
* **SimpleMailMessage**: lớp đại diện cho một tin nhắn email đơn giản.

9. Lấy dữ liệu từ 2 bảng bằng cách nào trong Spring Boot?

* Trong Spring Boot, để lấy dữ liệu từ 2 bảng,có thể sử dụng JPA (Java Persistence API) và Hibernate.
* Đầu tiên, cần tạo các entity class cho các bảng trong cơ sở dữ liệu. Sau đó, sử dụng annotation để định nghĩa mối quan hệ giữa các bảng.
* Sử dụng Spring Data JPA để tạo các repository cho các entity class. Với Spring Data JPA, có thể định nghĩa các phương thức truy vấn để lấy dữ liệu từ cơ sở dữ liệu một cách dễ dàng.

10. Xử lý 1 request ở Spring MVC như thế nào?

**Luồng thực thi:**

* + - Người dùng điền tên **website** cần truy cập trên **brower** 🡪 nhấn **Enter**
    - gửi **request** lên **server** (nơi triển khai ứng dụng **Spring**)
    - **DispatcherServlet** tiếp nhận **request**
    - chuyển tới **controller** tương ứng(thông qua các cơ chế **mapping –** ánh xạ đường dẫn)
    - **controller** nhận yêu cầu và gọi phương thức thích hợp (**GET** / **POST**)
    - **controller** tiến hành **render** dữ liệu và trả về tên **view** cho **DispatcherServlet**.
    - **DispatcherServlet** sẽ nhận trợ giúp từ **ViewResolver** để xác định **view** cho các yêu cầu.
    - **DispatcherServlet** chuyển dữ liệu tới **view** đó 🡪 chuyển tới trình duyệt.

11. (Trending) Spring Security là gì. Sử dụng như thế nào. Phân quyền như thế nào.

**Security** (**bảo mật**) – là sự hạn chế khả năng lạm dụng tài nguyên và tài sản. Gồm cơ chế bảo mật và phân quyền.

- **Spring Security** – là một **Framework** của **Spring**, dành riêng cho việc thiết lập bảo mật ứng dụng bao gồm **Authentication** và **Authorization**.

- **Authentication** (**xác thực**) – là 1 cơ chế / hành động nhằm thiết lập hoặc chứng thực 1 thông điệp / đối tượng là đáng tin cậy.

- **Authorization** (**phân quyền** / **cấp quyền**) – là quá trình xác định xem 1 đối tượng / người dùng có quyền truy cập vào 1 tài nguyên cụ thể nào đó để thực hiện 1 số hành động **CRUD**.

12. Cách sử dụng DI bằng interface, tại sao chỉ dùng interface mà lại gọi được class implement?

* **Dependency Injection (DI)** là một kỹ thuật trong lập trình phần mềm, cho phép các đối tượng được cung cấp các phụ thuộc của chúng thông qua các tham chiếu (hoặc interface), thay vì cứng thông qua việc khởi tạo các đối tượng bên trong chính nó.
* Khi sử dụng DI, chúng ta thường sử dụng các interface để định nghĩa các phụ thuộc của đối tượng mà không cần biết cụ thể là các lớp implement của interface đó. Khi cần sử dụng các đối tượng được cung cấp bởi DI, chúng ta chỉ cần gọi các phương thức được định nghĩa trong interface đó, và DI sẽ tự động cung cấp các lớp implement của interface để thực hiện các phương thức đó.=> đc gọi là **class implement**

51. Những lợi thế khi dùng Spring Boot so với khi không dùng?

1. **Tiết kiệm thời gian:** Spring Boot cung cấp cho người dùng một cách tiếp cận đơn giản và nhanh chóng để xây dựng các ứng dụng Java, giúp tiết kiệm thời gian so với việc tạo một ứng dụng từ đầu.
2. **Dễ dàng sử dụng:** Spring Boot là một framework rất dễ sử dụng với cấu hình mặc định hợp lý và tự động cấu hình cho phép người dùng chỉ tập trung vào logic xử lý chính của ứng dụng.
3. **Tiêu chuẩn hóa:** Spring Boot đảm bảo tiêu chuẩn hóa trong cách xây dựng các ứng dụng Java. Vì vậy, các ứng dụng được xây dựng bằng Spring Boot có thể dễ dàng chuyển đổi giữa các ứng dụng khác nhau mà không phải lo lắng về các vấn đề liên quan đến cấu hình.
4. Hỗ trợ tốt cho các tính năng như RESTful web services, cơ chế dependency injection, hệ thống quản lý trạng thái, caching, security, logging, v.v...
5. **Dễ dàng tích hợp:** Spring Boot cho phép tích hợp dễ dàng với các công nghệ khác, chẳng hạn như JPA, Hibernate, Thymeleaf, AngularJS, v.v..., giúp giảm thời gian phát triển và tăng khả năng tái sử dụng của các thành phần.
6. **Cộng đồng lớn:** Spring Boot là một trong những framework được ưa chuộng nhất trong cộng đồng Java, vì vậy người dùng có thể dễ dàng tìm kiếm và sử dụng các thư viện và plugin hỗ trợ.

52. Spring batch là gì?

* Spring Batch là một framework được phát triển bởi Spring để xử lý các batch job (công việc định kỳ) trong các ứng dụng Java. Batch job thường là các công việc được thực hiện định kỳ, với số lượng dữ liệu lớn, ví dụ như việc thực hiện đồng bộ hóa dữ liệu giữa các hệ thống, xử lý và lưu trữ các bản ghi trong các hệ thống dữ liệu lớn, v.v...
* Spring Batch có thể tích hợp dễ dàng với các công nghệ khác của Spring như Spring Core, Spring Data và Spring Integration, giúp tạo ra các ứng dụng batch job đáng tin cậy, linh hoạt và dễ bảo trì.

53. Spring khác gì Spring Boot?

* Spring là một framework phổ biến trong lập trình ứng dụng Java, giúp phát triển các ứng dụng nhanh chóng và dễ dàng hơn bằng cách cung cấp một nền tảng lập trình mạnh mẽ và linh hoạt. Spring cung cấp rất nhiều tính năng và module để giải quyết các vấn đề phổ biến trong lập trình như dependency injection, quản lý transaction, tích hợp với các công nghệ khác, v.v...
* Spring Boot là một công cụ được phát triển trên nền tảng Spring, giúp giảm thiểu sự phức tạp khi phát triển ứng dụng Spring bằng cách cung cấp một số tính năng mặc định và tự động cấu hình để giảm thiểu thời gian và công sức khi phát triển.

54. Validation spring thì ntn? Áp dụng chỗ nào là hiệu quả nhất.?

**Validation** – thực hiện đánh giá / xác minh tính hợp lệ của dự liệu đầu vào (đối với mọi ứng dụng web, dữ liệu luôn quan trọng hơn công nghệ)

1. Các bước áp dụng Spring Validation như sau:

* Định nghĩa lớp model: Định nghĩa lớp model để đại diện cho đối tượng dữ liệu cần xác thực.
* Thêm các annotation: Sử dụng các annotation của Spring Validation để xác định các ràng buộc và thông tin xác thực cho các thuộc tính của lớp model.
* Sử dụng Validator: Tạo một đối tượng Validator để xác thực dữ liệu đầu vào. Validator sẽ kiểm tra dữ liệu đầu vào và trả về các thông báo lỗi nếu có.
* Xử lý lỗi: Xử lý các thông báo lỗi trả về từ Validator để hiển thị cho người dùng hoặc để lưu vào cơ sở dữ liệu.

1. Cách áp dụng Spring Validation hiệu quả nhất là áp dụng trực tiếp trên lớp model, định nghĩa các ràng buộc để kiểm tra các thuộc tính của đối tượng dữ liệu khi người dùng nhập vào hoặc khi truyền vào trong các method của ứng dụng.

55. (Trending) Liệt kê một vài Annotation trong Spring?

**@NotNull** – không đc phép null, request gửi lên không đc mất trường.

**@NotEmpty** – không đc trống / empty (có thể áp dụng với trường collection).

**@NotBlank** – không đc phép null / khoảng trắng (áp dụng cho chuỗi).

**@Size** – kiểm tra độ dài min / max.

**@Min** / **@Max** – giá trị nhỏ nhất / lớn nhất.

**@Email** – định dạng email.

**@Pattern** – truy định dạng thức của 1 chuỗi.

**@DateTimeFormat** – định dạng ngày, giờ.

56. Cấu hình Hibernate?

1. Thêm thư viện Hibernate vào dự án: Bạn cần thêm các thư viện Hibernate vào dự án của mình, bao gồm cả Hibernate Core và Hibernate Entity Manager.
2. Định nghĩa cấu hình Hibernate: Bạn cần định nghĩa các thông số cấu hình cho Hibernate như tên cơ sở dữ liệu, username và password, đường dẫn đến các file mapping, và các tham số liên quan đến caching.
3. Định nghĩa entity class: Entity class là các đối tượng trong Hibernate tương ứng với các bảng trong cơ sở dữ liệu. Bạn cần định nghĩa các entity class trong Hibernate và ánh xạ các thuộc tính của chúng với các cột trong bảng tương ứng.
4. Sử dụng Hibernate API để tương tác với cơ sở dữ liệu: Bạn có thể sử dụng các API của Hibernate để thực hiện các thao tác tương tác với cơ sở dữ liệu như lưu trữ, cập nhật, xóa, tìm kiếm và truy vấn dữ liệu.

57. @Controller và @RestController khác nhau như thế nào? @RestController dùng để làm gì?

- **Ý nghĩa của Controller:**

Trong mô hình **MVC**, **controller** là nơi nhận **request** từ người dùng, xử lý **request**, xây dựng dữ liệu cho **view** (**model**) và chọn **view** để trả lại kết quả của cho người dùng.

- **@Controller** – dùng để đánh dấu 1 **class** là 1 **Spring MVC controller** (dùng cho **web application**).

- **@RestController** – dùng đánh dấu 1 **class** là **resfull controller** (dùng cho ứng dụng **web service**).

58. @Aspect dùng để làm gì?

* @Aspect là một annotation trong Spring Framework được sử dụng để triển khai việc lập trình hướng khía cạnh (Aspect-oriented programming – AOP.
* @Aspect cho phép bạn xác định các khía cạnh và lọc các phương thức hoặc các gói cần được áp dụng. Bằng cách sử dụng các annotation như @Before, @After, @Around, @AfterReturning và @AfterThrowing, bạn có thể triển khai các tác vụ như ghi log, kiểm soát quyền truy cập, kiểm tra thời gian thực thi, v.v.

59. Native query trong Java viết như thế nào? Transaction trong Java dùng để làm gì và đặt nó ở đâu?

* Native query trong Java là cách sử dụng các câu lệnh SQL trực tiếp trong code Java để truy vấn cơ sở dữ liệu. Để viết một native query trong Java, bạn có thể sử dụng annotation @Query của Spring Framework và cung cấp câu lệnh SQL trong giá trị của annotation đó.
* Transaction trong Java được sử dụng để quản lý các thao tác liên quan đến cơ sở dữ liệu, đảm bảo tính toàn vẹn của dữ liệu. Các thao tác đó bao gồm thêm, sửa, xoá, truy vấn dữ liệu, và một số thao tác khác liên quan đến cơ sở dữ liệu.

60. Đăng nhập với spring thì xây dựng như nào, làm sao để lưu pass?

1. Để đăng nhập với Spring bằng Java, chúng ta có thể sử dụng Spring Security. Đây là một framework phổ biến và mạnh mẽ để quản lý bảo mật trong ứng dụng web.
2. Để xây dựng chức năng đăng nhập với Spring bằng Java, có thể làm theo các bước sau:

* Tạo một đối tượng User lưu trữ thông tin người dùng, bao gồm tên đăng nhập (username), mật khẩu (password) và các thông tin khác nếu cần.
* Tạo một repository để truy vấn đối tượng User từ cơ sở dữ liệu.
* Tạo một service để thực hiện các thao tác liên quan đến đăng nhập.
* Xác thực đăng nhập trong controller.
* Nếu đăng nhập thành công, controller sẽ lưu thông tin người dùng vào session để sử dụng cho các hoạt động sau này.
* Để lưu trữ mật khẩu của người dùng, bạn cần sử dụng một thuật toán mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu để đảm bảo tính bảo mật.

61. Controller có tiêm Repository vào được ko, có bị lỗi ko?

Có thể tiêm Repository vào trong Controller. Việc này sẽ giúp bạn sử dụng các phương thức của Repository trong phương thức của Controller một cách dễ dàng hơn. Điều quan trọng là phải đảm bảo rằng bạn đã cấu hình đúng các bean cần thiết cho Repository trước khi sử dụng nó trong Controller.

62. saveAll() và saveAndFlush() trong JPA?

Trong JPA, **saveAll()** và **saveAndFlush()** là hai phương thức được sử dụng để lưu thay đổi vào cơ sở dữ liệu.

* **saveAll():** Phương thức này sẽ lưu một danh sách các đối tượng vào cơ sở dữ liệu. Nếu có lỗi xảy ra trong quá trình lưu, các đối tượng được lưu trước đó vẫn được lưu vào cơ sở dữ liệu và chỉ có đối tượng gây ra lỗi sẽ không được lưu.
* **saveAndFlush():** Phương thức này sẽ lưu một đối tượng vào cơ sở dữ liệu và đẩy các thay đổi vào cơ sở dữ liệu ngay lập tức. Nếu có lỗi xảy ra trong quá trình lưu, ngoại lệ sẽ được ném ra.

63.(Trending) Trình bày về DI và IoC?

1. **IoC** (**Inversion of Controller**) – “nguyên lý đảo ngược quyền điều khiển”, là **nguyên lý** trong phát triển phần mềm, trong đó việc điểu khiển các đối tượng hoặc thành phần của hệ thống đc thực hiện bởi **framework** hoặc các **container** (LTV không cần tạo đối tượng thay vào đó họ chỉ cần mô tả rằng nó sẽ được tạo như thế nào, **framework** sẽ lo phần còn lại).
2. **DI** (**Dependency Injection**) – là **design pattern** cho phép xóa bỏ sự phụ thuộc, là một cơ chế để triển khai **IoC**, trong đó thao tác tiêm (**inject**) vào các đối tượng khác được thực hiện bởi **container** hoặc **framework**.

(Giống với Nguyên tắc **D** trong **S.O.L.I.D** là **Dependency inversion principle** - **Nguyên lý đảo ngược phụ thuộc**: các **module** cấp cao không nên phụ thuộc vào các **module** cấp thấp, cả 2 nên phụ thuộc vào **abstraction**).

* Giảm sự phụ thuộc giữa các **module**, **object** giúp cho việc mở rộng **code** sau này được dễ dàng.
* Giúp cho việc quản lý và kiểm thử ứng dụng dễ dàng hơn.
* Giúp chúng ta tập trung vào việc viết **logic** nghiệp vụ của ứng dụng, còn việc tạo và quản lý các đối tượng đc giao cho **framework** lo.

**Có 3 cách triển khai:**

* **Constructor** - Dùng **inject** (tiêm) các **module** bắt buộc. Các **module** được tiêm nằm trong **constructor**, và được gán lần lượt vào các **field**.
* **Setter** method - Dùng **inject** các **module** tùy chọn. Mỗi **module** sẽ được **inject** thông qua **setter**, nằm ở tham số và cũng gán cho **field** nào đó.
* **Interface** (field) - Các **client** phải triển khai một **Interface** mà có một **setter** method dành cho việc nhận **dependency**.

64. Các thành phần cơ bản của 1 project Spring

* Spring Core: Đây là một module cốt lõi của Spring Framework, cung cấp các tính năng chính để tạo nên ứng dụng Spring. Nó bao gồm một bộ sưu tập các thành phần và API để thực hiện các nhiệm vụ như dependency injection, quản lý bean, xử lý lỗi, quản lý transaction, ...
* Spring MVC: Đây là một module trong Spring Framework, cung cấp một kiến trúc phát triển ứng dụng web, với hỗ trợ đầy đủ cho các tính năng như tạo và xử lý các yêu cầu HTTP, xử lý các form, quản lý session, ...
* Spring Data: Đây là một module Spring cung cấp các tính năng để tương tác với các hệ thống lưu trữ dữ liệu khác nhau, bao gồm cả SQL và NoSQL databases. Nó giúp đơn giản hóa việc tương tác với cơ sở dữ liệu và làm cho mã của bạn dễ bảo trì hơn.
* Spring Security: Đây là một module trong Spring Framework, cung cấp các tính năng bảo mật cho ứng dụng Spring. Nó bao gồm các tính năng như xác thực, phân quyền, kiểm tra giấy phép và nhiều tính năng khác để giúp bảo vệ ứng dụng của bạn.
* Spring Boot: Đây là một module Spring giúp tạo ra ứng dụng Spring một cách nhanh chóng và dễ dàng hơn. Nó cung cấp một số tính năng như auto-configuration, starter dependencies, và embedded web server để giúp đơn giản hóa quá trình phát triển ứng dụng.

65. Bean Scope? Default của Bean là gì?

- Bean là những module chính của chương trình, là 1 đối tượng được quản lý bởi Spring IoC container (do framework tự khởi tạo, tự quản lý vòng đời của đối tượng).

- Các cách khai báo:

+ Dùng annotation: @Controller, @Service, @Repository, @Component.

+ Dùng kết hợp @Configuration && @Bean.

+ Khai báo Bean bằng XML.

- Có 5 Bean Scope:

+ Singleton – chỉ duy nhất một thể hiện của bean sẽ được tạo cho mỗi container. Đây là scope mặc định cho spring bean.

+ Prototype – khác với Singleton, 1 thể hiện của bean sẽ được tạo cho mỗi lần được yêu cầu (request).

+ Request – giống với prototype scope, tuy nhiên nó dùng cho ứng dụng web, 1 thể hiện của bean sẽ được tạo cho mỗi HTTP request.

+ Session – mỗi thể hiện của bean sẽ được tạo cho mỗi HTTP Session.

+ Global-Session – được sử dụng để tạo global sesion bean cho các ứng dụng Portlet.

@Bean là gì: là một annotation trong Spring Framework được sử dụng để chỉ định rằng một phương thức trả về một đối tượng sẽ được quản lý bởi container IoC của Spring. Khi một phương thức được đánh dấu với @Bean, container sẽ tạo ra một đối tượng từ kết quả của phương thức đó và lưu trữ nó trong context của ứng dụng, để có thể được sử dụng và truy xuất bởi các phần khác của ứng dụng.

66. JPA là gì? Spring JPA là gì? Hibernate là gì?

- JPA (Java Persistence API) là 1 bộ đặc tả tiêu chuẩn của Java, trong đó mô tả các thao tác quản lý dữ liệu trong Java cung cấp các mô hình cho phép ánh xạ các bảng trong CSDL với các class.

- Spring Data Jpa là 1 phần của dự án Spring Data, giúp cải tiến JPA tiêu chuẩn, đơn giản hóa tầng truy xuất dữ liệu, tự tạo repository, tạo các truy vấn JPA thông quá tên phương thức.

- Hibernate là một ORM Framework mã nguồn mở giúp LTV viết ứng dụng Java có thể mapping các Object với hệ CSDL, và hỗ trợ thực hiện các khái niệm OOP với CSDL.

67. Làm sao để map đối tượng qua table bên DB? => sử dụng @Entity

68. Không dùng @Autowire thì tiêm service kiểu gì?

**Có 3 cách triển khai:**

+ **Constructor** - Dùng **inject** (tiêm) các **module** bắt buộc. Các **module** được tiêm nằm trong **constructor**, và được gán lần lượt vào các **field**.

+ **Setter** method - Dùng **inject** các **module** tùy chọn. Mỗi **module** sẽ được **inject** thông qua **setter**, nằm ở tham số và cũng gán cho **field** nào đó.

+ **Interface** (field) - Các **client** phải triển khai một **Interface** mà có một **setter** method dành cho việc nhận **dependency**.

69. Spring core gồm những phần nào?

* Dependency Injection (DI): là kỹ thuật để cung cấp các đối tượng cho các đối tượng khác thông qua việc tiêm (inject) vào các đối tượng cần sử dụng. Spring Core hỗ trợ DI thông qua các đối tượng Bean.
* Inversion of Control (IoC): là một kiểu thiết kế trong đó các đối tượng không tạo và quản lý các đối tượng khác mà thay vào đó chúng được cung cấp bởi một bên thứ ba. Spring Core cung cấp một Container cho phép quản lý các đối tượng Bean và cung cấp chúng cho các đối tượng cần sử dụng.
* Aspect-Oriented Programming (AOP): là một phương pháp lập trình để phân chia các tác vụ trên các lớp khác nhau trong ứng dụng. Spring Core cung cấp khả năng tạo ra các Aspect để phân chia các tác vụ, như log, caching, transaction management, ...
* Spring Expression Language (SpEL): là một ngôn ngữ biểu thức được sử dụng để cấu hình các giá trị cho các đối tượng trong Spring Framework.
* Resource Management: Spring Core hỗ trợ quản lý các tài nguyên như file, JDBC connections, JMS connections, ...
* Validation and Data Binding: Spring Core cung cấp các công cụ để thực hiện validation và data binding cho các đối tượng trong ứng dụng.
* Type Conversion: Spring Core hỗ trợ chuyển đổi kiểu dữ liệu giữa các đối tượng trong ứng dụng.
* Event Handling: Spring Core hỗ trợ xử lý các sự kiện trong ứng dụng.
* Internationalization: Spring Core hỗ trợ đa ngôn ngữ cho ứng dụng.

70. Phân quyền, Login, Mã hoá mật khẩu? Thuật toán nào mã hoá mật khẩu \* cho BCrypt trong Spring.?

* Authentication (xác thực): là 1 cơ chế / hành động nhằm xác thực user có phải là người dùng của ứng dụng.
* Authorization (phân quyền): là quá trình xác định xem user có quyền truy cập vào những trang nào.
* Thuật toán BCrypt được sử dụng để băm mật khẩu trong Spring. Thuật toán BCrypt tạo Chuỗi có độ dài 60. Để mã hoá mật khẩu bằng BCrypt trong Spring, sử dụng lớp BCryptPasswordEncoder.

71.Spring Boot là gì?

Spring boot là 1 trong số các module của Spring Framework, được sử dụng để tạo ra các ứng dụng độc lập có thể chạy ngay với rất ít cấu hình thêm.

Ưu điểm:

+ Tạo ứng dụng độc lập, chạy ngay bằng cách sử dụng lệnh Java.

+ Nhúng sẵn server Tomcat, Jetty.

+ Cung cấp sẵn các thư viện “starter”.

+ Tự động cấu hình Spring. + Không yêu cầu cấu hình XML.

72. Spring framework là gì?

* Spring Framework là một framework phát triển ứng dụng Java mạnh mẽ và phổ biến.
* Nó cung cấp các công cụ và thư viện cho việc phát triển các ứng dụng Java chuyên nghiệp và hiệu quả, giúp cho việc phát triển ứng dụng trở nên dễ dàng hơn và nhanh chóng hơn.
* Spring Framework có nhiều tính năng, bao gồm Dependency Injection, AOP (Aspect Oriented Programming), JDBC Abstraction, Transaction Management, MVC (Model-View-Controller) và nhiều thứ khác.
* Nó cho phép các nhà phát triển xây dựng các ứng dụng Java tốt hơn, với sự linh hoạt và khả năng mở rộng cao.

73. @Transactional?

* Trong Spring Framework, @Transactional là một annotation được sử dụng để định nghĩa một phương thức hoặc một class có thể thực hiện các hoạt động trong một transaction. Annotation này được sử dụng để quản lý transaction trong ứng dụng Spring.
* Khi một phương thức được đánh dấu với @Transactional, Spring sẽ tạo ra một transaction để bao quanh phương thức đó. Nếu phương thức thực hiện thành công, Spring sẽ commit transaction và lưu lại các thay đổi vào cơ sở dữ liệu. Nếu có lỗi xảy ra, Spring sẽ rollback transaction để đảm bảo tính toàn vẹn của cơ sở dữ liệu.

74. Các framework khác ngoài Spring Boot?

* JavaServer Faces (JSF): Đây là một framework Java được sử dụng để phát triển các ứng dụng web Java EE. Nó cung cấp một mô hình phát triển MVC (Model-View-Controller), hỗ trợ các thành phần UI và có tính năng điều hướng và validation.
* Struts: Đây là một framework Java EE được sử dụng để phát triển các ứng dụng web. Nó cung cấp một mô hình phát triển MVC (Model-View-Controller), hỗ trợ các tính năng điều hướng, validation và giao tiếp với cơ sở dữ liệu.
* Play Framework: Đây là một framework Java được sử dụng để phát triển các ứng dụng web. Nó được thiết kế để hỗ trợ việc phát triển các ứng dụng web theo kiểu Reactive Programming, với sự hỗ trợ của các tính năng như routing, templating và ORM.
* Vaadin: Đây là một framework Java được sử dụng để phát triển các ứng dụng web với giao diện người dùng đa dạng. Nó cung cấp các thành phần UI, hỗ trợ các tính năng điều hướng, validation và giao tiếp với cơ sở dữ liệu.
* Apache Wicket: Đây là một framework Java được sử dụng để phát triển các ứng dụng web. Nó cung cấp một mô hình phát triển component-based, với sự hỗ trợ của các tính năng như routing, templating và validation.

75. Nếu không muốn dùng @Service thì còn cách nào không ?

* Có thể sử dụng các annotation khác như @Component hoặc @Repository để thay thế cho @Service, nếu bạn không muốn sử dụng nó.
* Annotation @Component là annotation chung nhất trong số các annotation đánh dấu cho các class Spring. Nó có thể được sử dụng để đánh dấu bất kỳ class nào là một Spring component
* Annotation @Repository được sử dụng để đánh dấu các class làm việc với cơ sở dữ liệu.

76. Sử dụng Thymeleaf làm gì? Thymeleaf là gì?

* Thymeleaf là một thư viện template engine trong Java, được sử dụng để xây dựng các giao diện người dùng trong các ứng dụng web. Nó hỗ trợ việc tạo ra các mẫu HTML và XML dựa trên dữ liệu từ các model Java, và cho phép viết các biểu thức, điều kiện, vòng lặp, các đối tượng và các thuộc tính HTML, và các biểu thức thư viện.
* Một số tính năng chính của Thymeleaf bao gồm:
* Hỗ trợ xử lý dữ liệu theo mô hình MVC (Model-View-Controller).
* Hỗ trợ các tính năng động như biểu thức EL (Expression Language), lặp lại, điều kiện và thư viện tag.
* Hỗ trợ các tính năng thể hiện thông tin lỗi và thông tin từ các dữ liệu.
* Hỗ trợ kết hợp với các framework khác như Spring để xử lý các yêu cầu web và các dữ liệu tương tác.
* Đơn giản và dễ sử dụng.
* Thymeleaf là một công cụ mạnh mẽ để xây dựng các giao diện người dùng trong các ứng dụng web, và nó đã được sử dụng rộng rãi trong cộng đồng phát triển Java.