**3, Có bao nhiêu cách để thực hiện Dependency Injection? Trình bày?**

Dependency Injection (DI) là một cơ chế trong Spring Framework cho phép các đối tượng (objects) được chèn vào (injected into) một đối tượng khác, mà không cần phải tạo ra và quản lý các đối tượng đó trực tiếp. Cơ chế này giúp giảm sự phụ thuộc (coupling) giữa các thành phần của ứng dụng và tạo điều kiện cho việc quản lý, tái sử dụng và kiểm thử dễ dàng hơn.

Có 3 cách để thực hiện Dependency Injection:

Truyền Tham số vào Constructor

Truyền Tham số vào setter & getter

|  |  |
| --- | --- |
| 3  4  5  6  7  8  9  10  11  12  13  14 | class Teacher {  private Student student;  // Dựa vào constructor  Teacher (Student student) {  this.student = student;  }  // Dựa vào Setter method  void setStudent(Student student){  this.student = student;  } |

Thông qua Interface

|  |  |
| --- | --- |
| 1  2  3  4 | public class Teacher {  @Autowired  private Student student;  } |

Kết Luận: biết beans nào phụ thuộc vào beans nào để spring biết để nhúng các phụ thuộc đó vào. Lập trình viên chỉ quan tâm thực hiện còn spring container sẽ quản lý các bean.

4 Framework là gì ? Framework khác Library chỗ nào ?

Là một tập hợp các Library đã được đóng gói để hỗ trợ phát triển ứng dụng dựa trên framework đó. Đồng thời, Framework cung cấp các nguyên tắc, cấu trúc của ứng dụng mà chúng ta phải tuân thủ theo nó.

|  |  |
| --- | --- |
| **Framework** | **Library** |
| **Framework lớn hơn library**  **Nó như là một chương trình, khi sử dụng**  **- ---- Framework phải tuân thủ các nguyên tắc của Famework, sử dụng các.**  **Framework là khung chương trình, lập trình viên bổ sung code theo tuân thủ của famework để tạo ra ứng dụng.**  **Framework chủ động**  **Tập hợp các Library** | **Library là tập hợp các funtion hoặc các class hoặc các chức năng đc viết sẵn để thực hiện một nhiệm vụ nào đó.**   * **Library có thể sử dụng code trực tiếp mà không cần thay đổi cấu trúc code của dự án.** * **Library bị bộng** |

5 Spring Framework là gì ?

Spring Framework là một framework phát triển ứng dụng phổ biến trong lĩnh vực phát triển phần mềm Java. Nó cung cấp một môi trường phát triển toàn diện để xây dựng các ứng dụng di động, web và doanh nghiệp Java.

Là mã nguồn mở

Spring nhẹ trong suốt

Spring Framework được xây dựng dựa trên 2 nguyên tắc design chính là: [Dependency Injection](https://stackjava.com/design-pattern/dependency-injection-di-la-gi.html) và Aspect Oriented Programming.

6 Lợi ích của Spring Framework?

- có cơ chế dependency Injection (DI) và Inversion of Control (IoC) giúp giảm sự phụ thuộc (coupling) giữa các thành phần của ứng dụng.

7 Nguyên lý đảo ngược quyền điều khiển (Inversion of Control) là gì ?

Inversion of Control có thể hiểu là một nguyên lý thiết kế trong công nghệ phần mềm. Các kiến trúc phần mềm được được áp dụng thiết kế này sẽ được đảo ngược quyền điều khiển so với kiểu lập trình hướng thủ tục.

8. Bean là gì?

Bean là một đối tượng được quản lý bởi Spring Ioc container ( bộ điều khiển). Bean được đăng ký với Spring container hoặc đánh dấu băng các annotation.

Bean có các tinh chất như :

* Dependency Injection (DI)
* AOP (Aspect-Oriented Programming):
* Quản lý vòng đời (Lifecycle Management): Spring container quản lý vòng đời của các bean, tự động khởi tạo, hủy bỏ và xử lý các sự kiện liên quan đến vòng đời của bean.

10 @Autowire là gì?

@Autowired là một annotation được sử dụng để đánh dấu việc tự động liên kết (autowiring) các phụ thuộc (dependencies) vào một bean.

Ví dụ, nếu bạn có một lớp service cần sử dụng một repository để truy cập cơ sở dữ liệu, bạn có thể đánh dấu trường repository trong lớp service bằng @Autowired. Khi Spring Boot khởi tạo service, nó sẽ tự động tìm kiếm và liên kết một bean repository phù hợp và gán nó vào trường.

11, @Component có ý nghĩa gì?

@Component là một annotation trong Spring Framework có ý nghĩa đánh dấu một class là một thành phần (component) của ứng dụng. Khi một class được đánh dấu bằng @Component, Spring sẽ quản lý và tạo ra một instance (đối tượng) của class đó và đưa nó vào trong container của Spring, còn được gọi là application context.

* 13 Trình bày ý nghĩa của ModelAndView Class?
* **ModelAndView là sự kết hợp của 2 mục đích ModelMap and View . Nó cho phép controller trả về 1 giá trị bao gồm Model và View .**

1. Trình bày ý nghĩa của ModelMap class.?

* **Model là một interface trong khi đó ModelMap là một Class.**
* **ModelMap cài đặt lớp Map interface. Nên nó thêm các phương thức của Map.**

17 Phân biệt POST với PUT thường sử dụng để làm gì?

|  |  |
| --- | --- |
| POST | PUT |
| Thường tạo dữ liệu mới  Muốn thêm dữ liệu mới có ID tự tăng không trùng lặp thì dùng post | Cập nhật dữ liệu có sẵn |

18. @RequestMapping làm gì?

@RequestMapping là một annotation trong framework Spring MVC của Java, được sử dụng để ánh xạ các URL (Uniform Resource Locator) đến các phương thức xử lý tương ứng trong một Controller. Annotation này xác định các yêu cầu HTTP mà phương thức được xử lý, như GET, POST, PUT, DELETE, v.v.

Khi bạn đánh dấu một phương thức trong một Controller với @RequestMapping, bạn đang chỉ định rằng phương thức đó sẽ được gọi khi có một yêu cầu HTTP đến với URL tương ứng. Annotation này cho phép bạn xác định các thuộc tính như URL pattern, phương thức HTTP, kiểu dữ liệu trả về, kiểu dữ liệu đầu vào, v.v.

21 Thymeleaf là gì?

Thymeleaf là một Java template engine dùng để xử lý và tạo HTML, XML, Javascript, CSS và text.

22 Sử dụng lặp trong Thymeleaf như thế nào? Sử dụng điều kiện trong Thymeleaf như thế nào?

Thymeleaf cung cấp cho bạn vòng lặp **'each'**, và bạn có thể sử dụng nó thông qua thuộc tính (attribue) th:each. Đây là vòng lặp duy nhất được hỗ trợ trong Thymeleaf.

Th:if

Th:unless

Th:switch

Th:case

Trong ví dụ trên, phần tử **<p>** sẽ chỉ được hiển thị nếu **user.isAdmin** là **false**.

Bên cạnh **th:if** và **th:unless**, Thymeleaf còn cung cấp các cấu trúc điều kiện khác như **th:switch**, **th:case**, và **th:unless**, tương tự với các câu lệnh switch-case trong ngôn ngữ lập trình.

23 Formatter là gì ? Converter là gì ?

|  |  |
| --- | --- |
| Formatter | Converter |
| **Định dạng sang 1 định dạng khác** | **Chuyển đổi kiểu dữ liệu** |
|  |  |
|  |  |

24 **ORM là gì ?**

ORM (Object-Relational Mapping) là một kỹ thuật trong lập trình để ánh xạ dữ liệu giữa hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) và các đối tượng trong ứng dụng. Nó giúp giảm thiểu sự phụ thuộc trực tiếp vào ngôn ngữ truy vấn cơ sở dữ liệu (ví dụ: SQL) và cho phép lập trình viên làm việc với dữ liệu bằng cách sử dụng các đối tượng và phương thức tương tự như trong lập trình hướng đối tượng.

ORM được hỗ trợ thông qua Spring Data JPA. JPA (Java Persistence API) là một tiêu chuẩn ORM trong Java

**25 JPA là gì? Spring Data JPA là gì ?**

JPA (Java Persistence API) là một tiêu chuẩn ORM (Object-Relational Mapping) trong ngôn ngữ lập trình Java.

Cung cấp các hàng CRUD chỉ cần extends.

Và Spring Data JPA là được tầng được xây dự phía trên tầng JPA. Có nghĩa là nó sẽ sử dụng tất cả các chức năng của JPA như entity mapping, entity life-lifecycle, JPA query v.v để xây dựng một số tính năng như no-code repository, khởi tạo query thông qua tên hàm v.v. Những khái niệm này sẽ được giải thích ở phần sau.

**26 Hibernate là gì ? Hibernate làm gì với database?**

Hibernate là một thư viện ORM (Object Relational Mapping) mã nguồn mở giúp lập trình viên viết ứng dụng Java có thể map các objects (pojo) với hệ quản trị cơ sở dữ liệu quan hệ, và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cớ dữ liệu quan hệ.

27 Liệt kê một số annotation của hibernate?

1. @Entity: Đánh dấu một lớp Java là một thực thể (entity) trong cơ sở dữ liệu.
2. @Table: Xác định tên của bảng trong cơ sở dữ liệu tương ứng với một thực thể.
3. @Id: Đánh dấu một thuộc tính là khóa chính của thực thể.
4. @GeneratedValue: Xác định cách Hibernate sinh giá trị cho thuộc tính khóa chính tự động (như AUTO, IDENTITY, SEQUENCE, TABLE, ...).
5. @Column: Xác định thông tin về cột trong cơ sở dữ liệu tương ứng với một thuộc tính.
6. @OneToMany: Đánh dấu một mối quan hệ "một-nhiều" giữa các thực thể.
7. @ManyToOne: Đánh dấu một mối quan hệ "nhiều-một" giữa các thực thể.
8. @JoinColumn: Xác định cột khóa ngoại để thiết lập mối quan hệ giữa hai thực thể.
9. @Transient: Đánh dấu một thuộc tính không cần được lưu trữ trong cơ sở dữ liệu.
10. @NamedQuery: Xác định một truy vấn được đặt tên (named query) trong Hibernate.
11. @Cascade: Xác định quy tắc xử lý cascading (cascade operations) cho mối quan hệ.
12. @Fetch: Xác định cách Hibernate truy xuất dữ liệu từ cơ sở dữ liệu (như EAGER, LAZY).

28 Entity là gì?

1. rong Hibernate, @Entity là một annotation được sử dụng để đánh dấu một lớp Java là một thực thể (entity) trong cơ sở dữ liệu. Một thực thể đại diện cho một bảng trong cơ sở dữ liệu và các đối tượng của lớp được ánh xạ vào bảng này.
2. Khi bạn đánh dấu một lớp với @Entity, Hibernate sẽ tạo một ánh xạ (mapping) giữa các thuộc tính của lớp và các cột trong bảng tương ứng trong cơ sở dữ liệu. Điều này cho phép bạn thực hiện các thao tác đọc, ghi, cập nhật và xóa dữ liệu trong cơ sở dữ liệu thông qua các đối tượng Java.

29 Persistence Context & Entity Manager là gì?

30 Câu lệnh truy vấn động là gì? Câu lệnh truy vấn tĩnh là gì?

|  |  |
| --- | --- |
| Dynamic query ( câu truy vấn động) | Static query (câu truy vấn tĩnh) |
| Được xây dựng trong quá trình thực thi chương trình dựa trên các điều kiện thay đổi. Bao gồm nhiều điều kiện thay đổi | Được complie lúc compile time, hạn chế số lần compile khi sử dụng, được tạo ra và không thay đổi trong quá trình thực thi hoặc rất ít. Thường dùng truy vấn tất cả dữ liệu hoặc một thông tin cụ thển nào đó. |