1. **Vẽ và trình bày cách Spring MVC xử lý request?**
2. Dispatcher Servlet: Đây là một Servlet đặc biệt trong Spring MVC, nhận yêu cầu từ client và chuyển tiếp nó cho các thành phần xử lý tiếp theo trong quy trình.
3. Handler Mapping: Handler Mapping là thành phần định tuyến yêu cầu đến các Controller tương ứng. Nó kiểm tra URL và xác định Controller phù hợp để xử lý yêu cầu.
4. Controller: Controller là thành phần chịu trách nhiệm xử lý yêu cầu. Nó nhận yêu cầu từ Handler Mapping, thực hiện các xử lý logic cần thiết và trả về kết quả cho Dispatcher Servlet.
5. Service: Service là thành phần chứa các logic xử lý nghiệp vụ của ứng dụng. Nó được gọi bởi Controller để thực hiện các tác vụ liên quan đến xử lý dữ liệu, tính toán và truy cập vào các thành phần dữ liệu khác như DAO.
6. DAO (Data Access Object): DAO là thành phần xử lý truy cập dữ liệu. Nó chịu trách nhiệm thực hiện các thao tác truy vấn và cập nhật dữ liệu vào cơ sở dữ liệu.
7. Database: Đây là nơi lưu trữ dữ liệu của ứng dụng. Cơ sở dữ liệu được truy cập thông qua DAO để lấy và cập nhật thông tin liên quan.

**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.

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

Trong Spring Boot, Controller là một thành phần quan trọng trong mô hình MVC (Model-View-Controller) để xây dựng các ứng dụng web. Ý nghĩa của Controller trong Spring Boot là xử lý yêu cầu HTTP từ client và phản hồi lại client thông qua các phương thức HTTP như GET, POST, PUT, DELETE, vv.

Controller trong Spring Boot có ý nghĩa chính là điều khiển và xử lý các yêu cầu HTTP từ client, thực hiện các tác vụ xử lý logic và trả về kết quả tương ứng cho client. Nó làm nhiệm vụ kết nối giữa View và Model trong mô hình MVC trong Spring Boot.

* **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 .**
* , lớp ModelAndView trong Spring Framework có ý nghĩa quan trọng trong việc đóng gói dữ liệu mô hình và thông tin hiển thị, giúp Controller truyền dữ liệu và điều hướng đến các View tương ứng trong quá trình xử lý yêu cầu và phản hồi trong ứng dụng web.

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.**
* **15 Trình bày ý nghĩa của ViewResolver Interface?**
* Ý nghĩa chính của ViewResolver là kết nối Controller với giao diện hiển thị (View) trong quá trình xử lý yêu cầu và phản hồi. Nó cung cấp cách thức để tìm và xác định View dựa trên thông tin từ Controller và thông tin từ yêu cầu của client.

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.

Xử lý 6 loại templates

Theo tiêu chuẩn html 5, là bộ xử lý view được sử dụng cho các ứng dụng web.

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.

JPA là tập hợp các interface được định nghĩa theo quy chuẩn để cho Hiberrate triển khai

Cung cấp các hàng CRUD chỉ cần extends.

Spring Data JPA: là phần mở rộng của Spring Framework, giúp hổ trợ sử dụng JPA dễ dàng hơn,

Đơn giản hoá việc truy xuất, tự động tạo repository, tự tạo các câu lệnh truy vấn thông qua tên phương thức,

Các phương thức:

Repository

CRDU

PadingAndSortTing

JPARepository

**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. |

**31 Trạng thái của Entity bao gồm những gì?**

New( mới tạo ra chưa lưu vào csdl).

Persistent (được ghi xuống CSDL và quản lý trong Persistence Context).

Detached (tạm thời tách ra khỏi Persistence Context)

Removed

**32 Có bao nhiêu loại mapping trong Hibernate?**

Có 3 loại:

* XML Mapping
* Annotation Mapping
* Java-based Configuration

**33 SessionFactory là gì?**

Là một interface giúp tạo ra session kết nối đến database bằng cách đọc các cấu hình trong Hibernate configuration.

SessionFactory chịu trách nhiệm khởi tạo và duy trì các kết nối đến cơ sở dữ liệu

SessionFactory là đối tượng nặng (heavy weight object) nên thường nó được tạo ra trong quá trình khởi động ứng dụng và lưu giữ để sử dụng sau này.

SessionFactory là một đối tượng luồng an toàn (Thread-safe) và được sử dụng bởi tất cả các luồng của một ứng dụng.

Mỗi một database phải có một session factory. Vì vậy, nếu bạn đang sử dụng nhiều cơ sở dữ liệu thì bạn sẽ phải tạo nhiều đối tượng SessionFactory. Giả sử ta sử dụng MySQL và Oracle cho ứng dụng Java của mình thì ta cần có một session factory cho MySQL, và một session factory cho Oracle.

**34 Validation dữ liệu là gì?**

Là quá trình kiểm tra tính đúng đắn của dữ liệu, nhằm kiểm tra dữ liệu vào đúng hay chưa theo quy tắc nào đó. Nhằm đảm bảo dữ liệu

**35 Trình bày cách triển khai validate dữ liệu trong Spring?**

**Cách 1: Xử dụng các Annotion như @Empty, @NotNull,@Zise**

**Cách 2: implement interface** Validator

Cach 3: Sử dụng @Validated annotation và BindingResult: trên controller

Cách 4: Sử dụng @Valid annotation và Method Validation:

Cách 5: Sử dụng @ExceptionHandler:

**36 Binding Result là gì?**

Là một đối tượng dùng để lưu trữ lỗi, chứa các phương thức xử lý lỗi để truy xuất các lỗi của chúng

* hasErrors(): Kiểm tra xem có lỗi nào xảy ra không.
* getAllErrors(): Trả về danh sách các lỗi xảy ra trong quá trình validation.
* getFieldErrors(): Trả về danh sách các lỗi xảy ra trên từng trường của đối tượng.
* getGlobalErrors(): Trả về danh sách các lỗi xảy ra không liên quan đến từng trường cụ thể.
* addError(): Thêm một lỗi vào BindingResult.
* **37 AOP là gì? AOP gồm những yếu tố nào?**

Aspect Orited Programming là kỹ thuật lập trình khía cạnh, phân tách chương trình thành các module riêng lẽ, phân biệt và không phụ thuộc lẫn nhau.

Các yếu tố chính:

Aspect:

* Core: chứa các nghiệp vụ chính
* Gross- cuting: nghiệp vụ phụ

Join point và point cut: join pont là các điểm chèn các advice, point cut là các biểu thức kiểm tra có khớp hay không, khớp sẽ thực hiện các advice.

advice: có 5 loại

After, before, after running, after throwing, Aroand

**42 Cookie là gì?**

**Là các tệp được người dùng truy cập tạo ra, lưu trữ trên trình duyệt người dùng, để ghi nhớ các tuỳ chọn trên trang web để cung cấp các nội dung phù hợp.**

**43 Phân biệt Session và Cookie?**

|  |  |
| --- | --- |
| Cookie | Session |
| Đều lưu lại dữ liệu trong quá trình làm việc  Thời gian sống là có cookie  Lưu ở client  Biến lưu có kiểu String  Cookie không phụ thuộc Session  Hạn hết bộ nhớ | Thời gian sống là 1 phiên làm việc  Biến lưu nhiều kiểu dữ liệu  Lưu ở sever  Session dựa trên Cookie do có JS SSIONID  Session phụ thuộc cookie  Không hạn chế bộ nhớ |

**45 Web Service là gì? Lấy ví dụ về web service?**

Một web service là cách giao tiếp giữa hai hoặc nhiều ứng dụng được phát triển cho các nền tảng khác nhau. Các service này độc lập với các trình duyệt và hệ điều hành qua đó dễ dàng giao tiếp và nâng cao hiệu suất để phù hợp với nhiều người dùng. Service này có thể như một hàm, một bộ sưu tập các tiêu chuẩn hoặc các giao thức được triển khai trên server. Nó là một giao tiếp giữa một client và server hoặc thông tin liên lạc giữa hai thiết bị thông qua kết nối mạng.

**46 Phân biệt Web Service và Web truyền thông?**

|  |  |
| --- | --- |
| **Webservie** | **Web truyền thống** |
| * **Không có giao diện** * **Truy cập là các ứng dụng** * **Độc lập không quan tâm đến nền tảng là gì** * **Dữ liệu là json, xml truy cập bằng http method**   **Ưu điểm**  **Tái sử dụng trên các ứng dụng khác nhau**  **ứng dụng phân tán**  **hoạt động trên nhiều nền tảng** | * **Có giao diện** * **Người dùng truy cập**   **Sever lỗi thì các nền tảng khác lỗi**  **Lỗ hỏng bảo mật** |

**47 SOAP là gì? RESTful là gì?**

48 jQuery là gì? Ajax là gì?

**Jquery:**

jQuery là một thư viện JavaScript phổ biến được sử dụng để tương tác và thay đổi nội dung của các trang web. Nó cung cấp các phương thức và hàm rút gọn để thao tác với HTML, CSS và JavaScript một cách dễ dàng và tiện lợi. jQuery giúp cho việc viết mã JavaScript trở nên ngắn gọn hơn và tương thích với nhiều trình duyệt khác nhau.

**Ajax**: là nhóm công nghệ phát triển web nhằm tạo ra web giàu tương tác, nhanh và mượt

Gồm html, css,js, xml, Dom, xml http request.

**55 Authentication là gì ? Authorization là gì ?**

Authentication xác minh danh tính của người dùng và đảm bảo tính hợp pháp,

i Authorization xác định và kiểm tra quyền và quyền truy cập của người dùng sau khi đã được xác thực

60. **Sự khác nhau giữa SpringBoot và SpringMVC là gì?**

|  |  |
| --- | --- |
| **Spring Boot** | **Spring MVC** |
| **Là các module**  **Sử dụng các quy ước để tự Cung cấp các cấu hình sẵn**  **Các thư viện có sẵn**  **Đơn giản hoá cấu hình gadle và maven**  **Cung cấp tính năng kiểm tra số liệu thống kê** | **Model and view controller**  **Cung cấp các tính năng có sẵn** |