**BỘ CÂU HỎI PHỎNG VẤN MODULE 4**

**SPRING MVC**

|  |  |
| --- | --- |
| STT | Câu hỏi |
| **Tổng quan Spring MVC** | |
| 1 | **Vẽ và trình bày cách Spring MVC xử lý request** |
|  |  |
| 2 | **Trình bày cơ chế Dependency Injection:**  Dependency Injection (DI) là một mô hình thiết kế phần mềm trong lập trình hướng đối tượng, giúp giảm sự phụ thuộc giữa các thành phần của hệ thống. Cơ chế này được sử dụng để cung cấp các dependency (phụ thuộc) cho một đối tượng thay vì để đối tượng đó tạo ra hay quản lý chúng trực tiếp.  Dưới đây là mô tả cơ chế Dependency Injection:  **Dependency (Phụ thuộc):** Đây là một đối tượng mà một đối tượng khác cần để hoạt động. Có thể là một đối tượng, một giao diện, hoặc thậm chí là một giá trị.  **Client Class (Lớp Khách hàng):** Đây là đối tượng sử dụng một dependency.  **Service Class (Lớp Dịch vụ):** Đây là đối tượng cung cấp dependency.  **Injector (Bơm phụ thuộc):** Là nơi quản lý và cung cấp dependency cho các đối tượng sử dụng chúng. |
| 3 | **Có bao nhiêu cách để thực hiện Dependency Injection? Trình bày?**  Cơ chế Dependency Injection có thể thực hiện theo ba cách chính:  **Constructor Injection (Phụ thuộc qua hàm tạo):** Dependency được chuyển vào qua các tham số của hàm tạo của lớp. Điều này giúp đảm bảo rằng khi một đối tượng mới được tạo, nó đã có tất cả các dependency cần thiết.  **Setter Injection (Phụ thuộc qua phương thức đặt giá trị):** Dependency được chuyển vào thông qua các phương thức đặt giá trị (setter methods). Điều này cho phép thay đổi dependency sau khi đối tượng đã được tạo.  **Interface Injection (Phụ thuộc qua giao diện):** Dependency được chuyển vào thông qua các phương thức của một giao diện được triển khai bởi client class. |
| 4 | **Framework là gì ? Framework khác Library chỗ nào ?**  Một framework là một bộ công cụ phần mềm được thiết kế để hỗ trợ và tăng cường quá trình phát triển phần mềm. Nó cung cấp một cấu trúc, quy tắc và các thành phần chung để giúp nhà phát triển xây dựng và triển khai ứng dụng một cách nhanh chóng. Framework thường chứa các thư viện, công cụ phần mềm, và quy tắc mà nhà phát triển có thể sử dụng để giảm bớt công việc lặp lại và tập trung vào logic chính của ứng dụng.  Framework:Framework quyết định quy luật luồng kiểm soát và nhà phát triển sử dụng framework phải tuân theo.Thường lớn và mạnh mẽ, được sử dụng để xây dựng toàn bộ ứng dụng hoặc hệ thống.  Library: là 1 thư viện tập hợp các mã nguồn có thể tái sử dụng, có thể sử dụng 1 cách tùy ý. Có thể nhỏ và chỉ tập trung vào một số chức năng cụ thể. |
| 5 | **Spring Framework là gì ?**  Spring Framework là một framework phát triển ứng dụng Java được thiết kế để giảm bớt sự phức tạp trong việc xây dựng và triển khai các ứng dụng doanh nghiệp Java. Được ra mắt lần đầu vào năm 2003, Spring đã trở thành một trong những framework phổ biến nhất trong cộng đồng phát triển Java. |
| 6 | **Lợi ích của Spring Framework?**  Spring Framework mang lại nhiều lợi ích cho việc phát triển ứng dụng Java. Dưới đây là một số lợi ích chính của Spring:   1. **Inversion of Control (IoC):** Sử dụng nguyên tắc IoC giúp giảm sự phụ thuộc giữa các thành phần của ứng dụng. Spring quản lý việc tạo và quản lý các đối tượng, làm cho mã nguồn trở nên linh hoạt và dễ bảo trì hơn. 2. **Dependency Injection (DI):** Spring hỗ trợ DI, giúp giảm sự phụ thuộc giữa các thành phần và làm cho việc thay đổi phụ thuộc trở nên dễ dàng hơn. Điều này cũng tạo điều kiện cho việc kiểm thử và tái sử dụng mã nguồn. 3. **Aspect-Oriented Programming (AOP):** AOP giúp tách rời các quy tắc chung (ví dụ: log, bảo mật) từ logic chính của ứng dụng. Điều này giảm sự lặp lại mã và làm cho quy tắc chung có thể được tái sử dụng dễ dàng. 4. **Lập trình hướng đối tượng (OOP):** Spring Framework thúc đẩy sử dụng lập trình hướng đối tượng, giúp tổ chức mã nguồn và tăng tính tái sử dụng. 5. **Model-View-Controller (MVC):** Spring cung cấp một framework MVC cho phát triển ứng dụng web. Điều này giúp tách biệt logic xử lý, hiển thị và quản lý các yêu cầu từ người dùng, làm cho mã nguồn trở nên dễ bảo trì. 6. **Data Access:** Spring cung cấp nhiều cách tiếp cận dễ sử dụng để tương tác với cơ sở dữ liệu, giúp đơn giản hóa việc thao tác với dữ liệu. 7. **Transaction Management:** Spring hỗ trợ quản lý giao dịch, đảm bảo tính nhất quán và an toàn trong quá trình thực hiện các thao tác cơ sở dữ liệu. 8. **Bảo mật:** Spring Security cung cấp các tính năng mạnh mẽ để quản lý bảo mật trong ứng dụng, bao gồm xác thực, ủy quyền và kiểm soát truy cập. 9. **Kiểm thử (Testing):** Spring Framework được thiết kế để dễ kiểm thử, với khả năng triển khai kiểm thử đơn vị và kiểm thử tích hợp. 10. **Dễ mở rộng (Extensibility):** Spring cho phép mở rộng và tùy chỉnh thông qua việc sử dụng các module và tiện ích mở rộng. 11. **Community Support:** Spring có một cộng đồng lớn, hoạt động tích cực và cung cấp tài liệu phong phú. Điều này giúp nhà phát triển dễ dàng tìm kiếm hỗ trợ và tài nguyên. |
| 7 | **Nguyên lý đảo ngược quyền điều khiển (Inversion of Control) là gì ?**  Nguyên lý Đảo Ngược Quyền Điều Khiển (Inversion of Control - IoC) là một nguyên lý quan trọng trong lập trình và thiết kế phần mềm, mà Spring Framework nổi tiếng sử dụng để quản lý việc tạo ra và quản lý đối tượng. IoC đảo ngược quyền kiểm soát từ ứng dụng hoặc module gọi đến container hoặc framework, để container chịu trách nhiệm quản lý các thành phần của ứng dụng.   1. **Dependency Injection (DI):** Trong DI, các dependency của một đối tượng được chuyển vào đối tượng đó từ bên ngoài, thường thông qua hàm tạo, các phương thức đặt giá trị, hoặc các cơ chế khác. IoC container quản lý và cung cấp các dependency này. Điều này giúp giảm sự phụ thuộc và làm cho việc thay đổi dependency trở nên dễ dàng hơn. 2. **Service Locator:** Trong mô hình Service Locator, một đối tượng (Service Locator) chịu trách nhiệm cung cấp các đối tượng hay dịch vụ cho các thành phần khác. Các thành phần không cần biết nơi chúng có thể tìm thấy các đối tượng hay dịch vụ mà chúng cần sử dụng. |
| 8 | **Giải thích IoC? Lợi ích của IoC ?**  IoC là một nguyên lý trong lập trình phần mềm mà quay ngược quyền kiểm soát về việc tạo ra và quản lý đối tượng. Thay vì module hoặc ứng dụng kiểm soát việc tạo ra và quản lý đối tượng của mình, nó chuyển trách nhiệm này cho một container hoặc framework bên ngoài. IoC giúp tạo ra sự linh hoạt, giảm sự phụ thuộc và làm cho việc quản lý các thành phần trở nên dễ dàng hơn.  Lợi ích:   1. **Giảm sự phụ thuộc (Reduced Dependency):** IoC giúp giảm sự phụ thuộc giữa các thành phần của ứng dụng. Các đối tượng không còn phải tạo ra hoặc quản lý các dependency của mình mà chúng có thể chấp nhận chúng từ bên ngoài. 2. **Tăng tính tái sử dụng (Increased Reusability):** Việc quản lý các dependency bởi IoC container giúp tăng khả năng tái sử dụng mã nguồn. Các đối tượng có thể được sử dụng lại trong nhiều ngữ cảnh khác nhau mà không cần sửa đổi mã nguồn. 3. **Giảm độ phức tạp (Reduced Complexity):** IoC giúp giảm độ phức tạp của mã nguồn bằng cách chuyển trách nhiệm quản lý đối tượng cho một thành phần bên ngoài, giảm thiểu mã boilerplate và tăng tính linh hoạt của hệ thống. 4. **Kiểm thử dễ dàng (Easy Testing):** IoC làm cho việc kiểm thử dễ dàng hơn bằng cách cho phép thay thế các dependency bằng các đối tượng giả mạo (mock objects) trong quá trình kiểm thử. 5. **Mở rộng dễ dàng (Easy Extensibility):** Các thành phần của ứng dụng có thể được thêm mới hoặc thay thế mà không làm ảnh hưởng đến các thành phần khác, do đó làm cho việc mở rộng trở nên dễ dàng. 6. **Quản lý vòng đời (Lifecycle Management):** IoC container có thể quản lý vòng đời của các đối tượng, giúp đảm bảo rằng các đối tượng được tạo ra và giải phóng một cách đúng đắn. |
| 9 | **Bean là gì?**  Bean thường được sử dụng để mô tả các đối tượng được quản lý bởi IoC container của Spring. Đối tượng bean trong Spring có thể là bất kỳ đối tượng Java nào, từ đối tượng đơn giản đến các đối tượng phức tạp.Các đối tượng bean thường được khai báo và quản lý trong file cấu hình Spring, hoặc thông qua các annotation trong mã nguồn Java. IoC container của Spring sau đó chịu trách nhiệm về việc tạo ra, cấu hình, và quản lý các đối tượng bean này. |
| 10 | **Trong Spring có bao nhiêu Bean Scope?**   1. Có 5 bean Scope 2. Singleton (Mặc định) 3. Prototype: 4. Request: 5. Session: 6. Global Session: |
| 11 | **@Autowire là gì?**  @Autowired là một annotation trong Spring Framework, được sử dụng để thực hiện Dependency Injection (DI) tự động cho các trường, phương thức, hoặc constructors trong các bean.  Khi bạn đánh dấu một trường (field), phương thức hoặc constructor của một bean với @Autowired, Spring IoC container sẽ tự động tiêm (inject) các dependency vào đó, giảm sự phụ thuộc và làm cho quá trình quản lý các dependency trở nên dễ dàng hơn. |
| 12 | **@Component có ý nghĩa gì?**  @Component là một annotation trong Spring Framework được sử dụng để đánh dấu một class là một "component" và được quản lý bởi Spring IoC container. Khi một class được đánh dấu với @Component, IoC container sẽ quản lý vòng đời của nó, tạo ra và cấu hình các instance của class đó, và làm cho nó trở thành một bean mà bạn có thể sử dụng trong ứng dụng của mình |
| **Spring Controller** | |
| 13 | **Trình bày ý nghĩa của Controller**  **Xử Lý Yêu Cầu (Request Handling):** Controller nhận và xử lý các yêu cầu từ người dùng. Các yêu cầu này có thể là các hành động như đăng nhập, đăng ký, tìm kiếm, và nhiều hoạt động khác.  **Giao Tiếp với Model:** Controller tương tác với Model để lấy thông tin hoặc cập nhật dữ liệu. Model thường biểu diễn dữ liệu và logic xử lý của ứng dụng.  **Quyết Định Loại View:** Controller quyết định loại View nào sẽ được hiển thị để trả về kết quả cho người dùng. View là phần giao diện mà người dùng sẽ nhìn thấy.  **Chuyển Dữ Liệu đến View:** Sau khi xử lý yêu cầu và tương tác với Model, Controller chuyển dữ liệu cần thiết đến View để hiển thị cho người dùng.  **Hướng Dẫn Luồng Kiểm Soát (Flow Control):** Controller quyết định luồng kiểm soát của ứng dụng, xác định xem yêu cầu sẽ được điều hướng đến đâu và làm gì. |
| 14 | **Trình bày ý nghĩa của ModelAndView Interface**  Dữ Liệu (Model): ModelAndView chứa một đối tượng Model, nơi bạn có thể đặt các thuộc tính (attributes) mà bạn muốn chuyển đến View. Thông thường, các thuộc tính này là dữ liệu mà Controller muốn hiển thị trên giao diện người dùng.  View Name (Tên View): Ngoài dữ liệu, ModelAndView còn chứa tên của View mà bạn muốn hiển thị. Nó giúp Controller xác định View nào sẽ được render cho người dùng. |
| 15 | **Trình bày ý nghĩa của ModelMap Interface**  Dữ Liệu (Model): ModelMap giữ một bản đồ các thuộc tính (key-value) mà bạn muốn truyền đến View. Các thuộc tính này thường là dữ liệu mà bạn muốn hiển thị hoặc sử dụng trong giao diện người dùng.  Tích Hợp với ModelAttribute: Nếu bạn sử dụng annotation @ModelAttribute trong các phương thức của Controller để bind dữ liệu giữa Form và đối tượng Model, ModelMap cũng được tự động kết hợp và sử dụng trong quá trình này. |
| 16 | **Trình bày ý nghĩa của ViewResolver Interface**  Xác Định Cách Tìm Kiếm View: ViewResolver xác định cách Spring MVC sẽ tìm kiếm và xác định file View cụ thể dựa trên tên View được trả về từ Controller. Có nhiều loại ViewResolver khác nhau có thể được cấu hình trong ứng dụng.  Chọn Ra View Đúng Đắn: Khi một Controller trả về tên của một View, ViewResolver sẽ quyết định View cụ thể để hiển thị. Điều này có thể bao gồm việc xác định đường dẫn đến file JSP, Thymeleaf template, hoặc bất kỳ loại View nào khác được sử dụng trong ứng dụng.  Hỗ Trợ Nhiều Loại View: Spring hỗ trợ nhiều loại View như JSP, Thymeleaf, FreeMarker, Velocity, và một số loại View khác. ViewResolver giúp kết nối giữa tên View được trả về từ Controller và định dạng cụ thể của file View. |
| 17 | **Định nghĩa URI với các phương thức khác nhau như GET, POST, PUT, PATH, DELETE**  URI (Uniform Resource Identifier) là một chuỗi ký tự để định danh duy nhất một nguồn tài nguyên trên Internet. URI bao gồm URL (Uniform Resource Locator) và URN (Uniform Resource Name). URL là một dạng của URI và thường được sử dụng để chỉ định vị trí cụ thể của một tài nguyên trên mạng.  Các phương thức HTTP như GET, POST, PUT, PATCH, và DELETE đóng một vai trò quan trọng trong giao thức HTTP và được sử dụng để thực hiện các hành động khác nhau trên tài nguyên được xác định bởi URI. Dưới đây là mô tả của mỗi phương thức và cách chúng tương tác với tài nguyên: |
| 18 | **Phân biệt POST với GET**  GET và POST là hai phương thức chính trong giao thức HTTP (Hypertext Transfer Protocol) được sử dụng để truyền thông tin giữa máy khách (client) và máy chủ (server). Dưới đây là sự phân biệt giữa GET và POST:  Get: giới hạn ký tự 0-255, kiểu dữ liệu ko đa dạng, ko bảo mật, có hesder  Post:đa dạng, bảo mật, có body |
| 19 | **Phân biệt POST với PUT**  Put: mỗi lần gửi lên sẽ lưu lại trạng thái và nếu gửi nhiều lần giống nhau thì sẽ tính là 1 lần  Post: mỗi lần gửi lên sẽ tạo mới và tính là 1 lần nếu gửi nhiều lần tính nhiều lần |
| 20 | **Thao tác với form trong ứng dụng Spring MVC**  Trong ứng dụng Spring MVC, thao tác với form thường bao gồm việc tạo form, truyền dữ liệu giữa trình duyệt và máy chủ, xử lý dữ liệu được gửi từ form, và hiển thị kết quả. |
| 21 | **@RequestMapping làm gì?**  @RequestMapping là một annotation trong Spring Framework được sử dụng để ánh xạ các yêu cầu HTTP (HTTP requests) tới các phương thức xử lý trong các Controller. Nó cho phép bạn xác định URL mà một phương thức cụ thể của Controller sẽ xử lý. |
| 22 | **Trình bày cơ chế Data Binding**  Data Binding là quá trình tự động ánh xạ giữa dữ liệu trong các đối tượng Java và các trường trong các biểu mẫu HTML hoặc các tham số trong yêu cầu HTTP. Trong ngữ cảnh của ứng dụng web Java, Data Binding thường được thực hiện giữa các đối tượng Java (Model) và các trang JSP hoặc các biểu mẫu HTML. |
| 23 | **Thuộc tính consumes trong các Request Mapping là gì ?**  Trong Spring Framework, thuộc tính consumes được sử dụng trong các annotation @RequestMapping để xác định các loại dữ liệu phương thức có thể xử lý (consume). Thuộc tính này giúp chỉ định các kiểu dữ liệu phương thức có thể chấp nhận từ yêu cầu HTTP. |
| **Thymeleaf** | |
| 24 | **Template là gì? Template Engine là gì? Thymeleaf là gì?**  Template:  một file chứa đựng mã HTML, CSS,JS để tạo diện người dùng, cho phép phát triển web dễ dàng quản lý và tái sử dụng giao diện người dùng.  Template Engine:  Template Engine là một công cụ (engine) có nhiệm vụ chấm dứt (render) các templates để tạo ra các trang web hoặc nội dung động, tạo ra kết quả cuối cùng mà người dùng sẽ nhìn thấy.  Thymeleaf:  Thymeleaf là một Template Engine được sử dụng chủ yếu trong môi trường Java và được tích hợp chặt chẽ với Spring Framework. Nó là một công cụ mạnh mẽ cho việc tạo và hiển thị giao diện người dùng trong ứng dụng web Java. |
| 25 | **Sử dụng lặp trong Thymeleaf như thế nào? Sử dụng điều kiện trong Thymeleaf như thế nào?** |
| **Data Binding** | |
| 26 | **Data Binding là gì?**  Data Binding là quá trình liên kết giữa dữ liệu và giao diện người dùng trong một ứng dụng, thường được sử dụng để đồng bộ hóa giữa dữ liệu được lưu trữ và giao diện người dùng. |
| 27 | **Formatter là gì ? Converter là gì ?**  Formatter là một thành phần trong Spring Framework được sử dụng để chuyển đổi giữa kiểu dữ liệu nguồn và kiểu dữ liệu đích trong quá trình binding dữ liệu.  Converter cũng là một thành phần chuyển đổi giữa các kiểu dữ liệu, nhưng thường được sử dụng ở mức độ rộng hơn và không dựa vào quá trình binding dữ liệu. |
| **ORM và JPA** | |
| 28 | **ORM là gì ?**  ORM là viết tắt của "Object-Relational Mapping", đây là một kỹ thuật trong phát triển phần mềm và quản lý cơ sở dữ liệu. ORM giúp ánh xạ (mapping) giữa các đối tượng trong mã nguồn của ứng dụng và các bảng trong cơ sở dữ liệu quan hệ.  Khi sử dụng ORM, các đối tượng Java (hoặc các ngôn ngữ lập trình khác) được liên kết trực tiếp với các bản ghi trong cơ sở dữ liệu mà không cần phải viết các câu truy vấn SQL trực tiếp. Thay vào đó, ORM tự động thực hiện ánh xạ giữa các đối tượng và bảng cơ sở dữ liệu, cũng như giữa các thuộc tính của đối tượng và các cột trong bảng. |
| 29 | **JPA là gì? Spring Data JPA là gì ?** |
| 30 | **Hibernate là gì ? Hibernate làm gì với database** |
| 31 | **Liệt kê một số annoutation của hibernate?** |
| 32 | **Entity là gì?** |
| 33 | **Persistence Context & Entity Manager là gì?** |
| 34 | **Câu lệnh truy vấn động là gì? Câu lệnh truy vấn tĩnh là gì?** |
| 35 | **Trạng thái của Entity bao gồm những gì?** |
| 36 | **Có bao nhiêu loại mapping trong Hibernate?** |
| 37 | **SessionFactory là gì?** |
| 38 | **Liệt kê ba trạng thái của đối tượng trong hibernate?** |
| **Validation** | |
| 39 | **Validation dữ liệu là gì?** |
| 40 | **Trình bày cách triển khai validate dữ liệu trong Spring** |
| 41 | **Binding Result là gì?** |
| **AOP** | |
| 42 | **AOP là gì? AOP gồm những yếu tố nào?** |
| 43 | **Các loại Advice?** |
| 44 | **JoinPoint là gì?** |
| 45 | **Nêu cách triển khai AOP mà bạn dùng trong dự án** |
| 46 | **Xử lý ngoại lệ trong Spring Web MVC như thế nào?** |
| **Session & Cookie** | |
| 47 | **Cookie là gì?** |
| 48 | **Phân biệt Session và Cookie** |
| 49 | **Nêu cách triển khai Session mà bạn dùng trong dự án** |
| **Web Service** | |
| 50 | **Web Service là gì? Lấy ví dụ về web service** |
| 51 | **Phân biệt Web Service và Web truyền thông** |
| 52 | **REST là gì? RESTful là gì?** |
| 53 | **jQuery là gì? Ajax là gì?** |
| 54 | **Nêu ví dụ về cách bạn triển khai Ajax trong dự án của bạn** |
| 55 | **@RequestHeader và @ResponseHeader có ý nghĩa gì** |
| **I18N** | |
| 56 | **I18N và I10N là gì?** |
| 57 | **Nêu cách triển khai I18N và I10N trong dự án Spring** |
| 58 | **Interceptor là gì?** |
| **Bảo mật** | |
| 59 | **Bảo mật là gì? Cơ chế bảo mật trong Spring như thế nào?** |
| 60 | **Authentication là gì ? Authorization là gì ?** |
| 61 | **Liệt kê các cơ chế xác thực** |
| 62 | **CRSF là gì** |
| 63 | **CORS là gì** |
| **Spring Boot** | |
| 64 | **Spring Boot là gì?** |
| 65 | **Sự khác nhau giữa SpringBoot và SpringMVC là gì?** |
| 66 | **Trong trường hợp bạn muốn tuỳ chỉnh cấu hình cho Spring Boot chỉ tìm kiếm các bean trong một package nhất định** |
| 67 | **Cấu hình cho Spring Boot Tìm kiếm các Bean ở nhiều package khác nhau bằng cách nào?** |
| 68 | **Trình bày Spring Security? @EnableWebSecurity làm gì?** |
| 69 | **Trình bày Spring Boot JPA? Cài đặt?** |
| 70 | **Để sử dụng interface JpaRepository cần cung cấp những thông tin nào?** |
| 71 | **Bạn có biết annotation @ConfigurationProperties? Trình bày?** |