**BỘ CÂU HỎI PHỎNG VẤN MODULE 4**

**JWT (Json web Token)** là một chuỗi mã hóa được gửi kèm trong Header của client request có tác dụng giúp phía server xác thực request người dùng có hợp lệ hay không. Được sử dụng phổ biến trong các hệ thống API ngày nay.

**JWT bao gồm 3 phần và ngăn cách nhau bởi dấu chấm.**

* **Header**
* **Payload**
* **Signature (chữ ký số )**

**Header:**

**- Loại token** (mặc định là JWT - Thông tin này cho biết đây là một Token JWT)

**- Thuật toán đã dùng để mã hóa** (HMAC SHA256 - HS256 hoặc RSA). {“alg”: “HS256”, “typ”: “JWT”}

**Payload là phần sẽ chứa nội dung server (dữ liệu) ta truyền lên server**. Ngoài ra nó còn chứa đựng các thông tin về tolkien như ngày hết hạn, ngày sinh ra token, subject, etc.

* **Signature là một chuỗi được mã hoá bao gồm các thông tin header + payload + chữ ký. Chữ ký Signature trong JWT là một chuỗi được mã hóa bởi header, payload cùng với một chuỗi bí mật theo nguyên tắc sau:**

**Hướng dẫn xây dựng ứng dụng Spring Security:** Luồng đi của ứng dụng mình như sau.

1/ User nhập vào username và password sau đó bấm login .

2/ Server sẽ nhận được request từ người dùng và chuyển tới controller tương ứng do ta cấu hình trong file configure của spring security .

3/ Controller sẽ gọi Service và Service sẽ gọi database để lấy thông tin authentication đúng không và role người dùng là gì?.

4/ Sau khi có thông tin đúng thì trả kết quả lại cho người dùng.

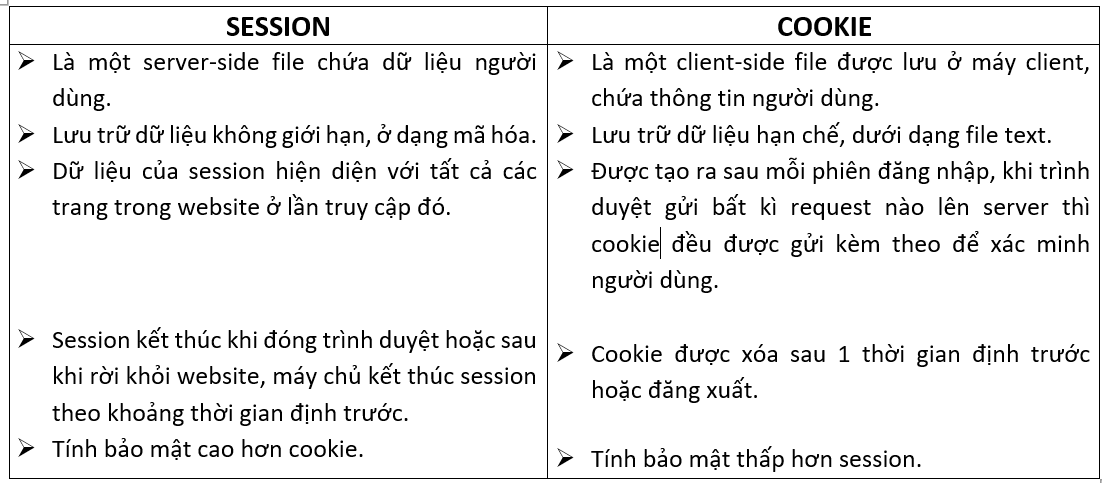
**Note: Phân biệt các @Controller**

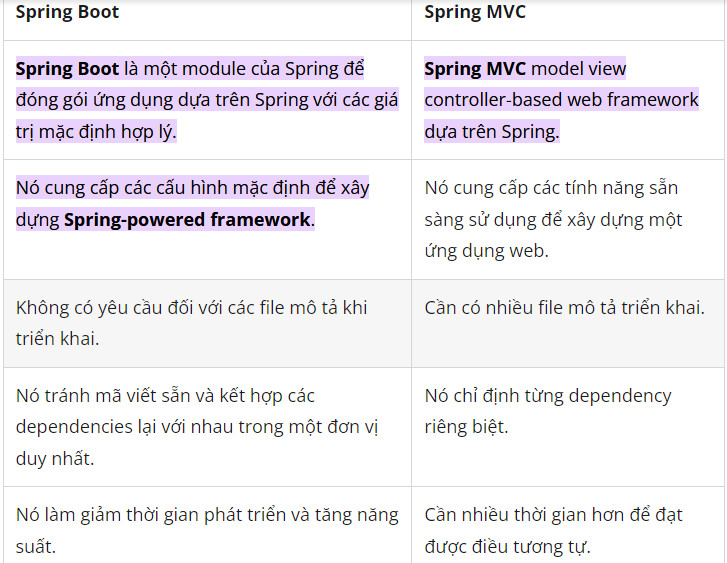
*- Controller là nơi nhận request từ người dùng, xử lý request, xây dựng dữ liệu cho view (model) và chọn view để trả lại kết quả của*

*cho người dùng*

|  |  |  |
| --- | --- | --- |
| **@Controller** | **@RestController** | **@ControllerAdvice** |
| @Controller ra đời trước  - dùng trong cả 2 project: 1. Project web, cụ thể là Spring MVC  2. Project web service  - Cần thêm @ResponseBody để return http response như JSON hoặc XML | @RestController ra đời sau nên nhà phát triển họ chủ động bỏ đi để code gọn hơn  - chỉ dùng trong project web service thôi  - không cần @ResponseBody vì đã tích hợp sẵn trong annotation | **-** Là trung tâm điều khiển, hành động, để liên kết các class hoặc các trường, trả về 1 body và đẩy các thông báo message ra  - Xử lý các ngoại lệ  - @ControllerAdvicelà một chuyên môn hóa của @Component chú thích cho phép xử lý các ngoại lệ trên toàn bộ ứng dụng trong một thành phần xử lý chung.  - Nó có thể được xem như một phương thức chặn các ngoại lệ được ném ra bởi các phương thức được chú thích @RequestMappingvà tương tự.   @ExceptionHandlerhoặc @InitBindercác @ModelAttributephương thức được chia sẻ trên nhiều @Controllerlớp. |







**Spring MVC** là một khung MVC hoàn chỉnh theo **định hướng HTTP được quản lý bởi Spring Framework và dựa trên Servlets**. Nó sẽ tương đương với JSF trong ngăn xếp JavaEE. Các yếu tố phổ biến nhất trong đó là các lớp được chú thích **bằng @Controller, nơi bạn triển khai các phương thức bạn có thể truy cập bằng các yêu cầu HTTP khác nhau. Nó có @RestController tương đương để triển khai các API dựa trên REST.**

**Spring boot** là **tiện ích giúp thiết lập ứng dụng nhanh chóng**, cung cấp cấu hình ngoài hộp để xây dựng các ứng dụng hỗ trợ Spring. Như bạn có thể biết, Spring tích hợp một loạt các mô-đun khác nhau trong ô của nó , như **spring-core , spring- data , spring-web (bao gồm Spring MVC, nhân tiện),** v.v. Với công cụ này, bạn có thể cho Spring biết bao nhiêu trong số chúng sẽ sử dụng và bạn sẽ có được thiết lập nhanh cho chúng (bạn được phép tự thay đổi nó sau này).

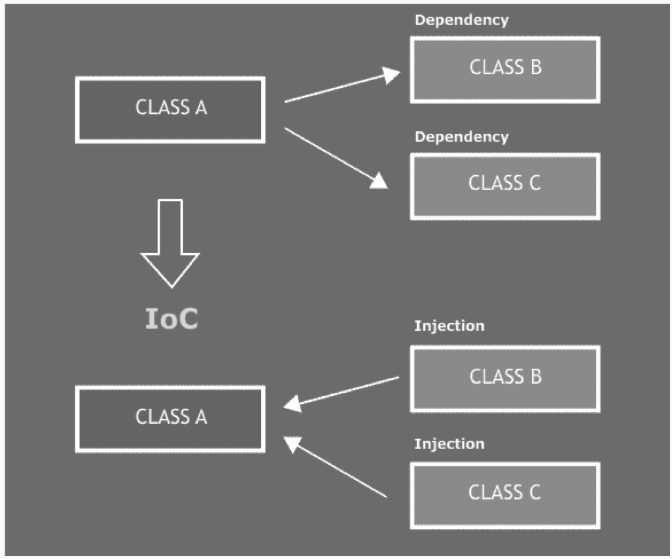
Vì vậy - **Spring MVC là một khung được sử dụng trong các ứng dụng web**

- **Spring boot là trình khởi tạo dự án dựa trên Spring sản xuất sẵn sàng**

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

*Inversion of Control (IoC) là một nguyên lý thiết kế trong công nghệ phần mềm trong đó các thành phần nó dựa vào để làm việc bị đảo ngược quyền điều khiển khi so sánh với lập trình hướng thủ thục truyền thống.*

*Với mô hình IoC thì class A không tự khởi tạo cũng không chịu trách nhiệm quản lý Class B, Class C. Nó nhận được hai dependency này từ bên ngoài thông qua một cơ chế nào đó (như bằng setter, bằng tham số hàm tạo lớp A, bằng gán thuộc tính ...)*



**8. Giải thích IoC? Lợi ích của IoC ?**

***- Inversion of Control (IoC)****là một nguyên lý thiết kế trong công nghệ phần mềm trong đó các thành phần nó dựa vào để làm việc bị đảo ngược quyền điều khiển khi so sánh với lập trình hướng thủ thục truyền thống.*

*\* LỢI ÍCH:*

*• Tách rời việc thực thi (execution) và việc triển khai (implementation)*

*• Dễ chuyển đổi giữa các implementation*

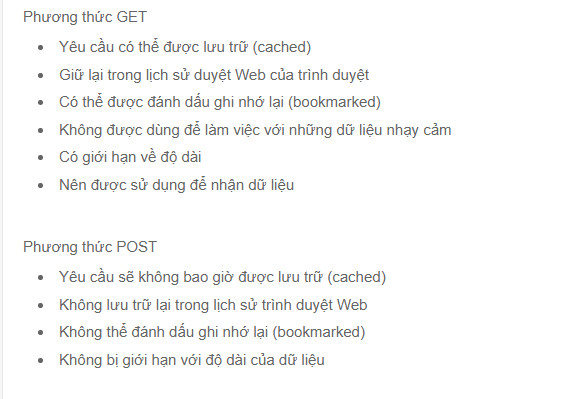
*• Dễ phân tách module hơn*

*• Dễ kiểm thử hơn, bằng cách tách rời các thành phần riêng lẻ*

*• Có thể triển khai IoC thông qua một số cơ chế: Strategydesign pattern, ServiceLocator pattern, Factory pattern, and* ***Dependency Injection.***

**NOTE: LỢI THẾ CỦA AJAX: tùy thuộc vào ngữ cảnh**

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| - Xác thực biểu mẫu  - Khả năng truyền dữ liệu  - Giảm lưu lượng máy chủ, thời gian phản hồi của 2 bên  - Không bắt buộc tải lại toàn bộ trang  - Giảm két nối đến server | - Bất kì ai cũng xem được mã nguồn (công nghệ mã nguồn mở)  - Công cụ tìm kiếm hạn chế  - Do có tính bảo mật nên: XMLHttpRequest bị hạn chế  - Nút “back” không có tác dụng |



|  |
| --- |
| **59. Bảo mật là gì? Cơ chế bảo mật trong Spring như thế nào?**  - Spring Security là bộ khung bảo mật ứng dụng Java web cung cấp cơ chế cấp phép quyền (authorization) và xác thực  người dùng (authentication).  \* Cơ chế bảo mật  - Xác thực (Authentication)  - Phân quyền (Authorization)  - Phân quyền hệ thống tệp tin  - Quản lý tài khoản  - Sử dụng SSL để truyền dữ liệu  - Mã hoá Firewalls |
| **60. Authentication là gì ? Authorization là gì ?**  - Authentication (xác thực) có nghĩa là xác nhận danh tính của riêng bạn.  - Authorization (ủy quyền) có nghĩa là cấp quyền truy cập vào hệ thống.  Nói một cách đơn giản, authentication là quá trình xác minh bạn là ai, trong khi authorization là quá trình xác minh những  gì bạn có quyền truy cập. |

|  |  |  |
| --- | --- | --- |
|  | **68. Trình bày Spring Security? @EnableWebSecurity làm gì?**  - Spring Security là một loạt các bộ lọc servlet giúp thêm xác thực và ủy quyền vào ứng dụng web.  - Spring Security cung cấp 2 cơ chế cơ bản:  + Authentication (xác thực): là tiến trình thiết lập một principal. Principal có thể hiểu là một người, hoặc một thiết bị, hoặc một hệ thống nào đó có thể thực hiện một hành động trong ứng dụng của bạn.  + Authorization (phân quyền) hay Access-control: là tiến trình quyết định xem một principal có được phép thực hiện một hành động trong ứng dụng của bạn hay không. Trước khi diễn tiến tới Authorization, principal cần phải được thiết lập bởi Authentication.  @EnableWebSecurity: <https://stackoverflow.com/questions/44671457/what-is-the-use-of-enablewebsecurity-in-spring> | |
|  | **2. Trình bày cơ chế Dependency Injection ?**  *- Trong kĩ thuật phầm mềm, Dependency injection là một công nghệ mà theo đó một object cung cấp sự phụ thuộc cho một object khác.*  *- Dependency là một object có thể được sử dụng (xem nó như 1 service).*  **-** *Thông thường, chúng ta chỉ thường gặp ba loại Dependency Injection sau:*  **+** *Constructor injection*  **+** *Setter injection*  **+** *Interface injection* | |
|  | **3.Có bao nhiêu cách để thực hiện Dependency Injection? Trình bày?**  *- Có 3 cách để thực hiện Dependency Injection:*  *+ Constructor injection: các dependency đc cung cấp thông qua constructor của class.*  *+ Setter injection: client tạo ra một setter method để các class khác có thể sử dụng chúng để cấp dependency.*  *+ Interface injection: dependency sẽ cung cấp một hàm injector để inject nó vào bất kì client nào đc truyền vào. Các client phải implement một interface mà có một setter method dành cho việc nhận dependency.* | |
|  | **4. Framework là gì ? Framework khác Library chỗ nào ?**  *- Framework là bộ "khung sườn" được cấu tạo thành các đoạn code có sẵn, hay nói cách khác nó là tập hợp các thư viện lập trình hoặc các API nhằm hỗ trợ một cách tối ưu. Giúp cho các lập trình ứng dụng, lập trình web trở nên nhanh chóng và dễ dàng hơn*.   |  |  | | --- | --- | | **Sự khác nhau giữa Framework và Library** | | | **Framework** | **Library** | | *- Có thể chứa nhiề Library bên trong nó.*  *- Cần thay đổi cấu trúc code trong dự*  *án đúng nguyên tác đề ra.*  *- Là khung chương trình, cho phép bổ sung các đoạn mã code.*  *- Mọi hoạt động trong Framework đều được diễn ra một cách chủ động.* | *- Là tập hợp con nằm trong Framework.*  *- Dễ dàng áp dụng các chức năng chương trình mà không cần thay đổi cấu trúc.*  *- Là nơi cung cấp các tính năng tiện ích và class sử dụng trong chương trình.*  *- Mọi hoạt động trong Library đều được diễn ra một cách bị động, khi ta cần và gọi đến chúng.* | | |
|  | **5. Spring Framework là gì ?**  *- Spring Framework là một framework ứng dụng mã nguồn mở, sử dụng để phát triển các ứng dụng doanh nghiệp.*  *- Cung cấp một mô hình đầy đủ cho việc phát triển và cấu hình các hệ thống java lớn.*  *- Nó giúp các nhà phát triển xây dựng những hệ thống và ứng dụng chạy trên JVM một cách đơn giản, tiện gọn, nhanh chóng và mềm dẻo* | |
|  | **6. Lợi ích của Spring Framework ?**   * *Spring framework được tổ chức theo kiểu module.* * *Dễ dàng kiểm tra và thử nghiệm.* * *Cung cấp API giúp thuận tiện trong việc dịch các ngoại lệ công nghệ cụ thể.* * *Hạn chế bộ nhớ và tài nguyên.* * *Được xây dựng theo mô hình MVC nên các tính năng thay thế web framework khác điều được cung cấp đầy đủ.* * Spring sẽ cung cấp được một giao diện quản lý transaction nhất quán. *Từ đó, có thể mở rộng được đến một local transaction cũng như mở rộng lên các global transaction.* | |
|  |  | |
|  |  | |
|  | **9. Bean là gì?**  *- Beans là những Java Object được quản lý bởi Spring Container.*  *- Khi sử dụng @Beans là đang khai báo 1 kiểu Object để cho Spring Container quản lý. Điều này giúp cho ứng dụng nhẹ hơn, dễ quản lý và mở rộng ứng dụng.* | |
|  | **10. Trong Spring có bao nhiêu Bean Scope?**  *- Singleton: Định nghĩa một thể hiện duy nhất của đối tượng trong mỗi spring IOC container.*  *- Prototype: Định nghĩa bất kì số các thể hiện đối tượng.*  *- Request: Định nghĩa vòng đời của mỗi HTTP request.*  *- Session: Định nghĩa vòng đời của mỗi HTTP session.*  *- Global session: Định nghĩa vòng đời của mỗi global HTTP session.* | |
|  | **11.@Autowire là gì?**  ***@Autowired****là một annotation được sử dụng để tiêm các đối tượng, đánh dấu cho Spring biết rằng sẽ tự động inject bean tương ứng vào vị trí được đánh dấu.*  *- Sau khi tìm thấy một class đánh dấu @Component thì quá trình inject Bean xảy ra theo cách như sau:*  *1. Nếu Class không có hàm Constructor hay Setter. Thì sẽ sử dụng Java Reflection để đưa đối tượng vào thuộc tính có đánh dấu @Autowired.*  *2. Nếu có hàm Constructor thì sẽ inject Bean vào bởi tham số của hàm*  *3. Nếu có hàm Setter thì sẽ inject Bean vào bởi tham số của hàm* | |
|  | **12.@Component có ý nghĩa gì?**  *\** ***@Component*** *định nghĩa một khuôn mẫu chung cho toàn bộ dự án.*  *@****Component là****một Annotation (chú thích) đánh dấu trên các Class để giúp****Spring****biết nó****là****một* ***Bean***  *\* Khi chạy, Spring chỉ tìm kiếm và chọn ra những class được đánh dấu với @Component để khởi tạo và đăng ký với ApplicationContext.*  *\* @Service, @Repository, @Controller là một trường hợp đặc biệt của @Component.* | |
|  | **13. Trình bày ý nghĩa của Controller**  *- 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* | |
|  | **14. Trình bày ý nghĩa của ModelAndView Interface**  *- Class ModelAndView có nhiệm vụ quan trọng trong việc tạo ra các hồi đáp (response). Các thể hiện của ModelAndView giúp dễ dàng tạo liên kết giữa data model và view. Trong đó data model là đối tượng chứa dữ liệu để hiển thị lên view.* | |
|  | **15. Trình bày ý nghĩa của ModelMap Interface**  *- Class ModelMap là lớp con của LinkedHashMap được sử dụng trong việc xây dựng data model bằng sử dụng kiểu dữ liệu cặp gồm key và value. ModelMap sử dụng phương thức addAttribute() để tạo liên kết giữa model và tên logic trong model. Thuộc tính đặt trong ModelMap có thể sử dụng để hiển thị dữ liệu dạng form trong sự kiện submit form.* | |
|  | **16. Trình bày ý nghĩa của ViewResolver Interface**  *- Các****ViewResolver****(trình phân giải view)****là****các đối tượng chịu trách nhiệm lấy các đối tượng View cho một hành động và vị trí cụ thể. Thông thường, các controller yêu cầu ViewResolvers chuyển tiếp đến một view với một tên cụ thể (một chuỗi được trả về bởi phương thức).*  *- ViewResolver cho phép ta đặt các thuộc tính như tiền tố (prefix), hậu tố (suffix) cho tên view để tạo URL trang view.*  **17. Định nghĩa URI với các phương thức khác nhau như GET, POST, PUT, PATH, DELETE**   * *URI cung cấp một kỹ thuật để xác định danh tính của một mục.* * *URI được sử dụng để phân biệt tài nguyên này với tài nguyên khác bất kể phương pháp được sử dụng.* * *URI không chứa đặc tả giao thức.* * *URI là "tập cha" của URL.* | |
|  | **18. Phân biệt POST với GET** | |
|  | **19. Phân biệt POST với PUT**  Điểm khác biệt giữa POST và PUT đó là các request PUT là như nhau.  Việc gọi một PUT request nhiều lần luôn trả về một kết quả y hệt trong khi gọi POST request lặp lại sẽ tạo ra cùng một tài nguyên nhiều lần. | |
|  | 20. Thao tác với form trong ứng dụng Spring MVC  1. Xử lý command object (tại thẻ form): th:object="${...}"  2. Inputs (tại thẻ input): th:field="\*{...}" (bind thuộc tính cho command object) | |
|  |  | |
|  | **21. @RequestMapping làm gì?**  *- Annotation RequestMapping dùng để ánh xạ một request tới một phương thức. Bạn có thể sử dụng*  *- @RequestMapping chú thích cho một phương thức hoặc một lớp. Một phương thức chú thích bởi annotation*  *- @RequestMapping sẽ trở thành phương thức xử lý request và sẽ được gọi khi dispatcher servlet nhận được một request với URL phù hợp.* | |
|  | **22. Trình bày cơ chế Data Binding**  ***- Data Binding****là****cơ chế****liên kết dữ liệu đầu vào hoặc đầu ra với các đối tượng model. ...* *Hay nói cách khác đó là sự kết nối dữ liệu của bean đặt trong model đến các điều khiển trên form.*  *- Khi tương tác với form, dữ liệu trên form sẽ được tự động chuyển đổi thành các thuộc tính của đối tượng liên kết với nó. Khi thay đổi dữ liệu của đối tượng thì dữ liệu trên các điều khiển cũng thay đổi theo* | |
|  | **23. Thuộc tính consumes trong các Request Mapping là gì ?**  *Sử dụng thuộc tính consumes để ánh xạ đến Content-Type của request.*  *Để quy định định dạng dữ liệu mà một request có thể được xử lí khi người dùng gửi request.* | |
|  | **24. Template là gì?**  *-* ***Template*** *là một cách thức thông dụng để hiển thị kết quả của một request về cho người dùng. Template định nghĩa một khuôn mẫu sẵn, sau đó ứng dụng sẽ đổ dữ liệu vào template để tạo ra một response thực tế và trả về cho người dùng.*  **- Template engine** là phần mềm được thiết kế để kết hợp các mẫu với một mô hình dữ liệu để tạo ra nhiều trang có cùng giao diện trên toàn bộ trang web  *-* ***Thymeleaf*** *là một Java template engine phía máy chủ, nó có thể làm việc với cả hai môi trường, web và non-web, nó có khả năng xử lý HTML, XML, JavaScript, CSS và thậm chí cả văn bản thuần túy.* | |
|  | **25. Sử dụng lặp trong Thymeleaf như thế nào? Sử dụng điều kiện trong Thymeleaf như thế nào?**  *- Khi sử dụng th:each, Thymeleaf cung cấp một cơ chế hữu ích cho việc theo dõi trạng thái của sự lặp lại: biến trạng thái (status variable).*  *- Các biến trạng thái được định nghĩa trong thuộc tính th:each và chứa các dữ liệu sau:*  *+ Chỉ số lặp hiện tại (iteration index), bắt đầu bằng 0. Đây là thuộc tính chỉ mục (index).*   * *+ Chỉ số lặp hiện tại (iteration index), bắt đầu bằng 1. Đây là thuộc tính đếm (count).* * *+ Tổng số phần tử trong biến lặp. Đây là thuộc tính kích thước (size).* * *+ Biến lặp cho mỗi lần lặp. Đây là thuộc tính hiện tại (current).* * *+ Cho dù lặp hiện tại (current iteration) là chẵn hay lẻ. Đây là những thuộc tính boolean even / odd.* * *+ Cho dù lặp hiện tại (current iteration) là lần đầu tiên. Đây là thuộc tính boolean first.* * *+ Cho dù lặp hiện tại (current iteration) là lần cuối cùng. Đây là thuộc tính boolean last.* * *Vòng lặp: <tr th:each="var : ${array}">* * *<td th:text=${var}></td></tr>* * *Điều kiện: <th:block th:if="${condition}">* * *<!-- khối lệnh được thực thi nếu điều kiện thoả mãn -->* * *</th:block>* | |
|  | **26. Data Binding là gì? (k học)**  *DataBinding là cơ chế liên kết dữ liệu đầu vào (hoặc đầu ra) với các đối tượng model*  *• DataBinding giúp cho việc tương tác với dữ liệu trở nên dễ dàng*  *• Sử dụng DataBinding,các form đều được liên kết với một đối tượng biểu diễn dữ liệu ở phía sau*  *• Khi tương tác với form, dữ liệu trên form sẽ được tự động chuyển đổi thành các thuộc tính của đối tượng liênkết với nó*  *• Data Binding hỗ trợ chuyển đổi dữ liệu (data conversion) và validate dữ liệu*  *• DataBinding trong Spring hoạt động dựa trên DataBinde* | |
|  | **27. Formatter là gì ? Converter là gì ?**  *- Formatter cũng hoạt động giống như converter, tức là chuyển đổi một kiểu dữ liệu sang kiểu dữ liệu khác.*  *- Tuy nhiên, kiểu dữ liệu nguồn của Formatter là String, trong khi đó converter có thể làm việc với bất cứ kiểu dữ liệu nguồn nào.*  *- Formatter phù hợp hơn với tầng web (web-tier), còn converter thì có thể sử dụng ở bất cứ tầng nào.*  *- Trong trường hợp muốn chuyển đổi dữ liệu nhập vào từ một trường input trong form trong một ứng dụng Spring MVC, chúng ta nên lựa chọn formatter hơn là converter* | |
|  | **28. ORM là gì ?**  *- ORM (Object-Relational Mapping) là kỹ thuật liên kết giữa các đối tượng trong lập trình với các đối tượng trong CSDL.*  *- Khi sử dụng lập trình hướng đối tượng và CSDL quan hệ, ORM sẽ giúp ánh xạ các lớp đến các bảng và các đối tượng đến các bản ghi trong các bảng.* | |
|  | **29. JPA là gì? Spring Data JPA là gì ?**  *- JPA là một bộ đặc tả tiêu chuẩn của Java, trong đó mô tả các thao tác quản lý dữ liệu quan hệ trong các ứng dụng sử dụng Java. JPA bao gồm:*   * *Bộ API, được định nghĩa trong gói javax.persistence* * *Java Persitence Query Language (Ngôn ngữ truy vấn Java Persistence)* * *Object/relational metadata*   *- Spring Data JPA là một module của Spring Framework. Mục đích của Spring Data JPA là giảm thiểu việc thực hiện quá nhiều bước để có thể implement được JPA.* | |
|  | **38. Liệt kê ba trạng thái của đối tượng trong hibernate?**  ***- detached****(tách rời) sau khi phiên được đóng bởi nhà phát triển. Các đối tượng tách rời sau đó có thể biến thành các đối tượng ổn định với các phương thức thích hợp được kích hoạt. Các đối tượng*  ***- persistent****(liên tục) là những đối tượng đang mở - chúng xảy ra bất cứ khi nào bạn lưu hoặc truy xuất một số trường hợp nhất định từ cơ sở dữ liệu Hibernate. Cá****c****đối tượng*  ***- transient****(không ổn định) là những “đối tượng lẻ” - vừa được tạo và chưa được phân biệt bởi một Phiên cụ thể.* | |
|  | **36. Có bao nhiêu loại mapping trong Hibernate?**  *4 loại:*   * *One-to-one* * *One-to-many* * *Many-to-one*   *Many-to-many* | |
|  | **32. Entity là gì?** Entity là một Java class tương ứng với một table trong database.  *- Entity là đối tượng đại diện cho dữ liệu ở trong ứng dụng, chẳng hạn Customer, Product, Student… có thể là các entity.*  *- Entity thường là POJO (Plain Old Java Object) – nghĩa là các lớp đơn giản, chỉ dùng để mô tả dữ liệu chứ không xử lý các logic nghiệp vụ.*  *- Entity sẽ được ánh xạ (mapping) tới một bảng trong CSDL. Chẳng hạn, lớp Customer có thể ánh xạ tới bảng customers trong CSDL, lớp Student có thể ánh xạ tới bảng students trong CSDL.*  **35. Trạng thái của Entity bao gồm những gì?** ***- Trạng thái của Entity bao gồm:******+ New (Mới tạo ra, chưa được lưu xuống CSDL)******+ Persistent (được ghi xuống CSDL và quản lý trong Persistence Context)******+ Detached (tạm thời tách ra khỏi Persistence Context)*** ***+ Removed: Mặc dù JPA chỉ cho phép xóa những entity đã được quản lý, Hibernate vẫn có thể xóa được những detached entity (nhưng phải thông qua method delete của session). Entity ở trạng thái removed thực ra chưa hoàn toàn bị xóa cho đến khi Session flush được gọi.*** | |
|  | **33. Persistence Context & Entity Manager là gì?** **Persistence Context** *Persistence Context là tập các thể hiện của entity được quản lý, tồn tại trong một kho dữ liệu. (Một entity ở trạng thái Persistence tương ứng với một dòng dữ liệu trong database và được quản lý bởi Persistence Context đang chạy).* ***Entity Manager*** *Entity Manager quản lý một đối tượng tham chiếu từ một thực thể, thực hiện đọc, ghi và xóa một thực thể.* | |
|  | **34. Câu lệnh truy vấn động là gì? Câu lệnh truy vấn tĩnh là gì?**  *- Câu lệnh truy vấn tĩnh là các câu lệnh không thay đổi tại runtime.*  *- Câu lệnh truy vấn động là câu lệnh được dựng tại thời điểm runtime, ví dụ như ứng dụng cho phép người dùng nhập truy vấn của chính mình.*  Điểm khác biệt và các giới hạn của hai loại truy vấn có thể tìm hiểu thêm ở đây:  https://www.geeksforgeeks.org/difference-static-dynamic-sql/ | |
|  |  | |
|  |  | |
|  | **37. SessionFactory là gì?**  ***- Session Factory Là****một interface giúp tạo ra session kết nối đến database bằng cách đọc các cấu hình trong một file xml và mỗi loại Database khác nhau sẽ có một cấu hình khác nhau. Từ đó, mỗi lần cần query hay insert dữ liệu, chúng ta sẽ tạo ra Session và sử dụng* | |
|  |  | |
|  | **39. Validation dữ liệu là gì?**  *- Validation là thao tác cho phép ứng dụng kiểm tra dữ liệu nhập vào bởi người dùng để đảm bảo tính hợp lý và chính xác khi xử lý các thao tác nghiệp vụ.*  *- Chẳng hạn, khi cần nhập địa chỉ email, ứng dụng cần đảm bảo email đó là hợp lệ.*  *Validation có thể được thực hiện ở các tầng khác nhau, chẳng hạn:*   * *- Validate ở tầng giao diện: Sử dụng JavaScript để validate* * *- Validate ở tầng back-end: Validate ở Controller/Service* * *- Validate ở tầng CSDL: Sử dụng các ràng buộc trong CSDL (NOT NULL, UNIQUE...)* | |
|  | **40. Trình bày cách triển khai validate dữ liệu trong Spring ? *Validation sử dụng interface Validator của Spring***  *1. Thêm thư viện hibernate-validator hay spring-boot-starter-validation vào dependencies.*  *2. Thêm các annotations như @NotNull, @NotEmpty, @Email vào các trường cần validate trong model.*  *3. Thêm annotation @Valid hay @Validated vào các phương thức nhận các đối tượng cần validate làm tham số.*  *Cách khác: custom validation bằng cách implements Validator/ ConstraintValidator* | |
|  | **41. Binding Result là gì?**  *- BindingResult] là đối tượng của Spring mà****giữ kết quả xác thực và ràng buộc và chứa các lỗi có thể xảy ra.***  ***-*** *BindingResult phải đến ngay sau khi đối tượng mô hình được xác thực nếu không xác thực đối tượng và ném ngoại lệ.*  *- Là một interface chỉ định cách thức đối tượng lưu trữ các kết quả của việc xác thực (validation) và truy xuất kết quả của việc xác thực.*  *- Kế thừa từ interface Errors.* | |
|  | **42. AOP là gì? AOP gồm những yếu tố nào? (ko)**  - Aspect Oriented Programming (AOP) – lập trình hướng khía cạnh: là một kỹ thuật lập trình (kiểu như lập trình hướng đối tượng) nhằm phân tách chương trình thành cách moudule riêng rẽ, phân biệt, không phụ thuộc nhau.  Các yếu tố quan trọng của AOP:  • Core concerns: hàm chính của chương trình (các method cần thực hiện log)  • Crosscutting concerns: những chức năng khác của chương trình: (chức năng log)  • Join points: một điểm của chương trình, là nơi có thể chèn những “custom action” của bạn  • Pointcut: có nhiều cách để xác định joinpoint, những cách như thế được gọi là pointcut.  • Advice: là những xử lý phụ được thêm vào xử lý chính, code để thực hiện các xử lý đó được gọi Adivce.  Tham khảo: https://viblo.asia/p/aop-la-gi-uu-nhuoc-diem-cua-aop-aspect-oriented-programming-trong-java-3P0lPk2oZox | |
|  | **43. Các loại Advice?**  *• Before Advice: thực thi trước join point.*  *• After Returning Advice: thực thi sau khi joint point hoàn thành bình thường*  *• After Throwing Advice: thực thi nếu joint point kết thúc do có ngoại lệ được tung*  *• After (finally) Advice: thực thi sau khi joint point hoàn thành bất kể dưới điều kiện nào*  *• Around Advice: thực thi cả trước và sau joint point* | |
|  | **44. JoinPoint là gì? (ko)**  *Điểm nối, là những vị trí xác định trong luồng thực thi của chương trình mà tại đó một khía cạnh sẽ bắt đầu tham gia vào. Điểm gia nhập có thể là một lời gọi phương thức, một lệnh tung expection, hay một lệnh chỉnh sửa thuộc tính ...*  *- Một vị trí nào đó trong luồng thực thi chương trình mà advice sẽ được weave (đan vào).*  *- Các join point phổ biến bao gồm: vị trí đọc/ghi giá trị của field, vị trí gọi/thực thi một phương thức/constructor, hay vị trí mà một ngoại lệ được tung ra.* | |
|  | **45. Nêu cách triển khai AOP mà bạn dùng trong dự án** | |
|  | **46. Xử lý ngoại lệ trong Spring Web MVC như thế nào?**  *Có bốn phương thức để xử lý ngoại lệ:*   * *2.1. Sử dụng try-catch. Phương pháp này thường để xử lý các ngoại lệ tại request (Controller).* * *2.2. Sử dụng @ExceptionHandler để ủy quyền cho front controller xử lý ngoại lệ.* * *2.3. Sử dụng HandlerExceptionResolver. Phương pháp này được dùng để xử lý ngoại lệ tại servlet.* * *2.4. Sử dụng trang báo lỗi. Phương pháp này được dùng để bắt các fatal error hay các ngoại lệ sinh ra ngoài phạm vi của Spring.* | |
|  | **47. Cookie là gì?**  *- Khái niệm****cookie****cũng được dùng để lưu những thông tin tạm thời. Nhưng tập tin cookie sẽ được truyền từ server tới browser và được lưu trữ trên máy tính của bạn khi bạn truy cập vào ứng dụng.*  *Cookie là một mẫu thông tin được lưu trữ bởi server trên trình duyệt web của người dùng.*  *Server thiết lập cookie trong khi gửi lại response cho một request từ trình duyệt web. Trình duyệt lưu trữ web các cookie và gửi lại chúng với request tiếp theo đến cùng một máy chủ.*  *Cookie thường được sử dụng để quản lý phiên, theo dõi người dùng và lưu trữ các tùy chọn của người dùng.* | |
|  |  | |
|  |  | |
|  |
| **51. Phân biệt Web Service và Web truyền thông** | |
| **52. REST là gì? RESTful là gì?**  *REST là viết tắt của Representational State Transfer (chuyển trạng thái đại diện) là một kiểu kiến trúc lập trình.*  *Nó định nghĩa các quy tắc để thiết kết các web service chú trọng vào tài nguyên hệ thống*  *Trong kiến trúc REST mọi thứ đều được coi là tài nguyên. Nó có thể ở dạng hiển thị, chẳng hạn như XML, JSON hoặc HTML.*  *Trong RESTful web services, trạng thái của tài nguyên quan trọng hơn hành động được thực hiện đối với tài nguyên.*  **RESTful***API***là***một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource.* | |

|  |  |
| --- | --- |
| **53. jQuery là gì? Ajax là gì?**  *Jquery: JQuery là một thư viện JavaScript gọn nhẹ (lightweight), "viết ít hơn, làm nhiều hơn" (write less, do more). Mục đích của jQuery là giúp sử dụng JavaScript dễ dàng hơn trên trang web của bạn. JQuery thực hiện rất nhiều công việc phổ biến mà đòi hỏi phải viết nhiều dòng mã JavaScript, và đưa chúng vào các phương thức mà bạn có thể gọi chỉ với một dòng mã.*  *JQuery cũng đơn giản hoá rất nhiều những thứ phức tạp của JavaScript, như gọi AJAX và thao tác DOM*  *AJAX là viết tắt của JavaScript và XML không đồng bộ (Asynchronous JavaScript and XML). AJAX là một kỹ thuật mới để tạo ra các ứng dụng web tốt hơn, nhanh hơn và tương tác hơn với sự trợ giúp của XML, HTML, CSS và Java Script.*  *\* Một số tính năng:*  *+ Điều khiển HTML/DOM.*  *+ Điều khiển CSS.*  *+ Các phương thức sự kiện HTML.*  *+ Các hiệu ứng và hoạt hình (animation).*  *+ AJAX.*  \* AJAX (Asynchronous JavaScript and XML) là một kỹ thuật mới để tạo ra các ứng dụng web tốt hơn, nhanh hơn và tương tác hơn với sự trợ giúp của XML, HTML, CSS và Java Script.  \* AJAX sử dụng XHTML cho nội dung, CSS cho hình thức, cùng với Document Object Model và JavaScript để hiển thị nội dung động. | |
| **54. Nêu ví dụ về cách bạn triển khai Ajax trong dự án của bạn (ko)**   1. *Từ trình duyệt của client, ta có một sự kiện để gọi ajax. Khi đó javascript sẽ tạo nên một đối tượng XMLHttpRequest. Và đối tượng này sẽ được gửi một request đến server ví dụ: Khi người dùng click vào ô input và chọn skill tìm việc làm của người dùng. Khi đó ta sẽ lấy thông tin đó và gửi đến server và cần trả về các việc làm tương ứng phù hợp với người dùng.* 2. *Khi server nhận được HttpRequest từ đó sẽ xử lý request và trả về response cho web. Server xử lí lấy ra các*[*việc làm javascript*](https://topdev.vn/viec-lam-it/javascript--kt22)*chẳng hạn thuộc skill mà người dùng cần và trả về dữ liệu của các việc làm.* 3. *Sau khi nhận được một response từ server, Javascript sẽ xử lý và cập nhật vào trang web cho chúng ta.* | |
| **55. @RequestHeader và @ResponseHeader có ý nghĩa gì**  *Header request parameters là các thông số kèm theo khi gửi yêu cầu lên server*  *Header response parameters là các thông số kèm theo khi server gửi kết quả về cho client.*  *Request headers chứa thông tin của tài nguyên được lấy về, hoặc thông tin về khách hàng request tài nguyên đó.*  *Response headers chứa các thông tin bổ sung cho response, như địa điểm hoặc về server cung cấp nó.*  *Theo sau các annotations @RequestHeader hay @ReponseHeader, ta dùng thêm tên các trường tương ứng để lấy các thông tin cần thiết từ request/response headers. Ta vẫn có thể lấy toàn bộ request/response headers bằng việc khởi tạo nó dưới dạng Map<String, String> headers* | |
|  | **56. I18N và I10N là gì?**  *I18n - Quốc tế hóa - Cung cấp một số loại khung để bạn có thể dễ dàng trao đổi các chuỗi, đồ họa, âm thanh và các tài nguyên khác và thường xử lý các quy ước giao tiếp khác nhau.*  *L10n - Bản địa hóa - Quá trình tạo ra một chuỗi các chuỗi, đồ họa, âm thanh, vv để bạn có thể nhắm mục tiêu một quốc tịch, ngôn ngữ hoặc khu vực cụ thể.* | |
|  | **58. Interceptor là gì?**  \* Trong Spring, khi một request được gửi đến controller, trước khi request được xử lý bởi Controller, nó phải vượt qua các Interceptor (0 hoặc nhiều).  Spring Interceptor là một khái niệm khá giống với Servlet Filter.  Spring Interceptor chỉ áp dụng đối với các request đang được gửi đến một Controller.  \* Ví dụ: Sử dụng interceptor để thêm request header trước khi gửi request đến controller và thêm response header trước khi gửi response về client. | |
|  |  | |
|  |  | |
|  | **61. Liệt kê các cơ chế xác thực**  - Http Basic Authentication  - Cookies  - Signatures  - AWS Request Signing Flow  - One-time Passwords | |
|  | **62. CRSF là gì**  CSRF (Cross Site Request Forgery) là kỹ thuật tấn công bằng cách sử dụng quyền chứng thực của người dùng đối với một website. Hiểu một cách nôm na, đây là kỹ thuật tấn công dựa vào mượn quyền trái phép. | |
|  | **63. CORS là gì**  CORS là một cơ chế cho phép nhiều tài nguyên khác nhau (fonts, Javascript, v.v…) của một trang web có thể được truy vấn từ domain khác với domain của trang đó. CORS là viết tắt của từ Cross-origin resource sharing.  Về cơ bản thì từ phía server sẽ thông báo cho trình duyệt biết là server chỉ chấp nhận resquest từ origin nào và những phương thức HTTP nào.  Khó hiểu quá, mời mọi người tham khảo thêm tại đây: https://topdev.vn/blog/cors-la-gi/  https://topdev.vn/blog/cors-la-gi/ | |
|  | **64. Spring Boot là gì?**  - Spring Boot là một web framework mã nguồn mở được xây dựng dựa trên kiến trúc microservice. Framework Spring Boot cung cấp một môi trường có thể hoạt động ngay nhưng đồng thời nó cũng cho phép lập trình viên tuỳ chỉnh cấu hình thông qua mã dựng sẵn trong bộ mã cua nó. | |
|  | **66. Trong trường hợp bạn muốn tuỳ chỉnh cấu hình cho Spring Boot chỉ tìm kiếm các bean trong một package nhất định**  Cách 1: @ComponentScan({"name.of.package1", "name.of.package2"})  Cách 2: @SpringBootApplication(scanBasePackages = {"name.of.package1", "name.of.package2"}) | |
|  |  | |
|  | **69. Trình bày Spring Boot JPA? Cài đặt?**  - Spring Data JPA, là một phần của đại gia đình Spring Data. Nó hỗ trợ việc mở rộng các repositories JPA.  - Spring Data JPA hỗ trợ tối ưu cho tầng truy cập dữ liệu JPA, giúp việc xây dựng ứng dụng Spring với công nghệ truy cập dữ liệu trở nên dễ dàng hơn. | |
|  | **70. Để sử dụng interface JpaRepository cần cung cấp những thông tin nào?**  public interface JpaRepository<T, ID>  + T: Entity/Model class name.  + ID: Kiểu dữ liệu ID của entity mà repository quản lý. | |
|  | **71. Bạn có biết annotation @ConfigurationProperties? Trình bày?**  @ConfigurationProperties cho phép truy xuất các thuộc tính được đặt trong các tệp cấu hình như application.properties hay application.yml một cách tự động và mạnh mẽ. | |