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     * Bất kỳ request nào tới ứng dụng web đều sẽ được gửi tới Front Controller (Dispatcher Servlet) * Front Controller sẽ sử dụng Handler Mapping để biết được controller nào sẽ xử lý request đó * Controller nhận request, gọi tới các class service thích hợp để xử lý yêu cầu. * Sau khi xử lý xong, Controller sẽ nhận được model từ tầng Service hoặc tầng DAO. * Controller gửi model vừa nhận được tới Front Controller (Dispatcher Servlet) * Dispatcher Servlet sẽ tìm các mẫu view, sử dụng view resolver và truyền model vào nó. * View template, model, view page được build và gửi trả lại Front Controller * Front Controller gửi một page view tới trình duyệt để hiển thị nó cho người dùng. |
| 2 | **Trình bày cơ chế Dependency Injection**  *Dependency Injection* nghĩa là các đối tượng (object) sẽ không cần quan tâm đến việc khởi tạo hay get các *phụ thuộc (dependency)* của mình, thay vào đó, sẽ có một bên thứ 3 đứng ra, *tiêm (inject)* các *dependency* này vào đối tượng. |
| 3 | Có bao nhiêu cách để thực hiện Dependency Injection? Trình bày? |
| 4 | **Framework là gì ? Framework khác Library chỗ nào ?**  Framework là các ứng dụng phần mềm có tính trừu tượng (abstraction), cung cấp các tính năng chung và thông dụng, có thể tuỳ biến để tạo nên những ứng dụng cụ thể khác nhau • Mỗi framework cung cấp một phương pháp riêng biệt để xây dựng và triển khai ứng dụng  Diểm khác biệt lớn nhất giữa Framework và Library đó chính là cơ chế “Inversion of Control” • Với Library: Ứng dụng nắm quyền điều khiển (control) • Với Framework: Framework nắm quyền điều khiển |
| 5 | **Spring Framework là gì ?**  **Spring framework** là một Java Platform mã nguồn mở, một giải pháp gọn nhẹ dành cho Java Enterprise. Với Spring Framework các nhà phát triển có thể tạo ra các mã có hiệu suất cao, dễ kiểm thử và có thể sử dụng lại được. |
| 6 | **Lợi ích của Spring Framework?**  • Framework giải quyết các vấn đề thông dụng, giúp lập trình viên tập trung vào xử lý nghiệp vụ • Giúp tăng tốc độ phát triển • Cung cấp môi trường làm việc tiêu chuẩn, giúp dễ giao tiếp giữa các bên khi cùng tham gia phát triển • Các framework thường có cộng đồng phát triển lớn, các giải pháp đã được đánh giá và thử nghiệm, hệ sinh thái đầy đủ giúp nhanh chóng xây dựng được các giải pháp tuỳ biến |
| 7 | **Nguyên lý đảo ngược quyền điều khiển (Inversion of Control) là gì** ?  IoC là một nguyên lý trong phát triển phần mềm, trong đó việc điều khiển các đối tượng hoặc các thành phần của hệ thống được thực hiện bởi framework hoặc các container • IoC cho phép framework nắm giữ quyền điều khiển luồng thực thi của hệ thống và gọi các mã nguồn khác |
| 8 | **Giải thích IoC? Lợi ích của IoC ?**  • 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 |
| 9 | **Bean là gì?**  Các đối tượng tạo thành xương sống của ứng dụng và được quản lý bởi Spring IoC container được gọi là **Bean**. Một bean là một đối tượng được khởi tạo, lắp ráp, và được quản lý bởi một Spring IoC container. |
| 10 | **Trong Spring có bao nhiêu 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. Khi sử dụng scope này cần chắc chắn rằng các bean không có các biến/thuộc tính được share. * **Prototype**: Một 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, một 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. |
| 11 | **@Autowire là gì?**  @Autowired là một annotation config của Spring, để sử dụng nó ta phải khai báo thẻ  <context:annotation-config /> trong file config. |
| 12 | @Component có ý nghĩa gì? |
|  | |
| 13 | **Trình bày ý nghĩa của Controller**  Spring MVC cung cấp các annotation như @Controller và @RestController để khai báo các controller Các annotation này thực hiện các nhiệm vụ như ánh xạ tới URL, khai báo tham số của request, xử lý ngoại lệ… |
| 14 | Trình bày ý nghĩa của ModelAndView Interface  • ModelAndView đại diện cho một view cùng với các dữ liệu sử dụng trong view đó • ModelAndView có thể kèm theo status của Response |
| 15 | Trình bày ý nghĩa của ModelMap Interface |
| 16 | **Trình bày ý nghĩa của ViewResolver Interface**  • ViewResolver là cơ chế để xử lý tầng view của Spring MVC • ViewResolver ánh xạ tên của view sang đối tượng view tương ứng |
| 17 | Định nghĩa URI với các phương thức khác nhau như GET, POST, PUT, PATH, DELETE |
| 18 | **Phân biệt POST với GET**   | GET | POST | | --- | --- | | Phương thức GET gửi thông tin người dùng đã được mã hóa được phụ thêm vào yêu cầu trang, truyền thông tin thông qua url. | Phương thức POST truyền thông tin thông qua HTTP header | | Dữ liệu của METHOD GET gửi đi thì hiện trên thanh địa chỉ (URL) của trình duyệt. | Dữ liệu được gửi đi với METHOD POST thì không hiển thị trên thanh URL | | HTTP GET có thể được cache bởi trình duyệt | HTTP POST không cache bởi trình duyệt | | HTTP GET có thể duy trì bởi lịch sử đó cũng là lý do mà người dùng có thê bookmark được. | HTTP POST không thể duy trì bởi lịch sử đó cũng là lý do mà người dùng không thê bookmark HTTP POST được. | | Không bảo mật | Bảo mật | | Thực thi nhanh hơn POST vì những dữ liệu gửi đi luôn được webbrowser cached lại. | Thực thi chậm hơn GET | | phương thức GET ứng với cùng một yêu cầu đó webbrowser sẽ xem trong cached có kết quả tương ứng với yêu cầu đó không và trả về ngay không cần phải thực thi các yêu cầu đó ở phía server. | Khi dùng phương thức POST thì server luôn thực thi và trả về kết quả cho client | | Phương thức GET được giới hạn gửi tối đa chỉ 2048 ký tự | Phương thức POST không có bất kì hạn chế nào về kích thước dữ liệu sẽ gửi. | | Không gửi được nhị phân. | Phương thức POST có thể sử dụng để gửi ASCII cũng như dữ liệu nhị phân. | |
| 19 | **Phân biệt POST với PUT**  Bảo toàn dữ liệu là dữ liệu cuối cùng và dữ liệu ban đầu là không đổi.  Vậy phương thức bảo toàn dữ liệu là phương thức thực hiện bao nhiêu lần đi chăng nữa thì kết quả cũng chỉ giống như thực hiện 1 lần.  Và PUT chính là phương thức bảo toàn dữ liệu nên khi chúng ta ấn submit bao nhiêu lần đi chăng nữa thì dữ liệu cũng chỉ tính như chúng ta submit 1 lần. Còn đối với POST thì chúng ta submit quá nhiều lần thì có thể nó sẽ cho ra những kết quả khác nhau. |
| 20 | Thao tác với form trong ứng dụng Spring MVC |
| 21 | **@RequestMapping làm gì?**  Annotation @RequestMapping được sử dụng để map request với class hoặc method xử lý request đó.  @RequestMapping có thể được áp dụng với controller class hoặc method trong controller class. |
| 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 • Data Binding giúp cho việc tương tác với dữ liệu trở nên dễ dàng • Sử dụng Data Binding, 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 liên kết với nó • Data Binding hỗ trợ chuyển đổi dữ liệu (data conversion) và validate dữ liệu • Data Binding trong Spring hoạt động dựa trên Data Binder |
| 23 | Thuộc tính consumes trong các Request Mapping là gì ?  consumes là định dạng dữ liệu khi user request tới |
| **Thymeleaf** | |
| 24 | **Template là gì? Template Engine là gì? Thymeleaf là gì?**  Thymeleaf là một Java template engine dùng để xử lý và tạo HTML, XML, Javascript, CSS và text.  • Thymeleaf là một bộ xử lý view được sử dụng cho các ứng dụng web và các ứng dụng độc lập. • Thymeleaf có thể xử lý HTML, XML, JavaScript, CSS và cả text. • Thymeleaf cung cấp một cách thức xây dựng template (mẫu) hiện đại và dễ bảo trì. • Thymeleaf được xây dựng phù hợp với các tiêu chuẩn của web, đặc biệt là HTML5. |
| 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? |
| 27 | **Formatter là gì ? Converter là gì ?**  - Spring Converter là một đối tượng được dùng để chuyển đổi kiểu dữ liệu này sang kiểu dữ liệu khác  - 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. |
| **ORM và JPA** | |
| 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 • ORM cho phép truy xuất dễ dàng đến dữ liệu thông qua các đối tượng lập trình • ORM giúp lập trình viên tập trung thao tác với các đối tượng, không cần quá quan tâm đến CSDL thực tế đang dùng |
| 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  Bởi vì JPA chỉ là một bộ các đặc tả, do đó để sử dụng được thì cần tới một framework triển khai các đặc tả đó, chẳng hạn như là Hibernate. |
| 30 | **Hibernate là gì ? Hibernate làm gì với database**  Hibernate framework là một giải pháp ORM (Object Relational Mapping) mã nguồn mở, gọn nhẹ. Hibernate giúp đơn giản hoá sự phát triển của ứng dụng java để tương tác với cơ sở dữ liệu.  Mã nguồn mở và nhẹ: Hibernate Framework là mã nguồn mở có giấy phép LGPL và nhẹ.  2. Hiệu suất nhanh: Hiệu suất của Hibernate Framework là nhanh bởi vì bộ nhớ cache được sử dụng trong nội bộ Hibernate Framework. Có hai loại bộ nhớ cache trong Hibernate Framework, gồm bộ nhớ cache cấp một và bộ nhớ cache cấp hai. Bộ nhớ cache cấp một được bật bằng lệnh mặc định.  3. Truy vấn cơ sở dữ liệu độc lập: HQL (Hibernate Query Language) là phiên bản hướng đối tượng của SQL. Nó tạo ra các truy vấn cơ sở dữ liệu độc lập. Vì vậy, bạn không cần phải viết các truy vấn cơ sở dữ liệu cụ thể. Trước Hibernate, nếu dự án có cơ sở dữ liệu bị thay đổi, chúng ta cần phải thay đổi truy vấn SQL dẫn đến sự cố bảo trì.  4. Tạo bảng tự động: Hibernate framework cung cấp phương tiện để tạo ra các bảng cơ sở dữ liệu tự động. Vì vậy, không cần phải tạo ra các bảng trong cơ sở dữ liệu bằng tay.  5. Đơn giản lệnh join phức tạp: Có thể lấy dữ liệu từ nhiều bảng một cách dễ dàng với Hibernate framework.  6. Cung cấp thống kê truy vấn và trạng thái cơ sở dữ liệu: Hibernate hỗ trợ bộ nhớ cache truy vấn và cung cấp số liệu thống kê về truy vấn và trạng thái cơ sở dữ liệu. |
| 31 | Liệt kê một số annoutation của hibernate? |
| 32 | **Entity là gì?**  Entity là đối tượng đại diện cho dữ liệu ở trong ứng dụng • Entity thường là POJO (Plain Old Java Object) • Entity sẽ được ánh xạ (mapping) tới một bảng trong CSDL • Trong JPA, một entity cần tuân thủ: • Được gắn với annotation javax.persistence.Entity • Có một constructor là public và không có tham số • Không được khai báo final • Các biến đối tượng cần được khai báo là private, protected hoặc ở mức package-private |
| 33 | **Persistence Context & Entity Manager là gì**?  Persistence Context là tập các thể hiện của entity được quản lý, tồn tại trong một kho dữ liệu • Interface EntityManager: • Khai báo các phương thức để tương tác với persistence context • Tạo hoặc xoá các thể hiện của entity • Tìm kiếm entity theo khoá chính • Thực thi các câu lệnh truy vấn lên entity |
| 34 | **Câu lệnh truy vấn động là gì? Câu lệnh truy vấn tĩnh là gì?**  Phương thức createQuery() của lớp EntityManager giúp tạo các câu truy vấn động (dynamic query) • Ví dụ: **public** List findWithName(String name) { **return** em.createQuery( **"SELECT c FROM Customer c WHERE c.name LIKE :custName"**) .setParameter(**"custName"**, name) .setMaxResults(10) .getResultList(); }  • Phương thức createNamedQuery() của lớp EntityManager giúp tạo các câu truy vấn tĩnh (static query) • Ví dụ, khai báo named query: • Sử dụng named query @NamedQuery( name="findAllCustomersWithName", query="SELECT c FROM Customer c WHERE c.name LIKE :custName" ) @ PersistenceContext public EntityManager em; ... customers = em.createNamedQuery("findAllCustomersWithName") .setParameter("custName", "Smith") .getResultList(); |
| 35 | Trạng thái của Entity bao gồm những gì? |
| 36 | Có bao nhiêu loại mapping trong Hibernate? |
| 37 | **SessionFactory là gì?**  **Session Factory** Là một interface giúp tạo ra session kết nối đến database bằng cách đọc các cấu hình trong Hibernate configuration. Mỗi một database phải có một session factory. Ví dụ nếu 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 |
| 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**?  [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 Spring sẽ không xác thực đối tượng và ném ngoại lệ.  Khi Spring thấy @Valid, nó cố gắng tìm trình xác nhận cho đối tượng được xác thực. Spring sẽ tự động chọn các chú thích xác thực nếu bạn đã bật tính năng điều khiển chú thích và hướng dẫn sử dụng. Sau đó, Spring gọi trình xác nhận và đặt bất kỳ lỗi nào trong BindingResult và thêm BindingResult vào mô hình xem. |
| AOP | |
| 42 | **AOP là gì? AOP gồm những yếu tố nào?**  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. |
| 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?**  • 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? |
| **Session & Cookie** | |
| 47 | **Cookie là gì?**  Cookie là một đoạn văn bản ghi thông tin được tạo ra và lưu trên trình duyệt của máy người dùng. Cookie thường được tạo ra khi người dùng truy cập một website, cookie sẽ ghi nhớ những thông tin như tên đăng nhập, mật khẩu, các tuỳ chọn do người dùng lựa chọn đi kèm. Các thông tin này được lưu trong máy tính để nhận biết người dùng khi truy cập vào một trang web.  **Session**dịch là **phiên làm việc**. Nó là cách đơn giản để lưu trữ 1 biến và khiến biến đó có thể tồn tại từ trang này sang trang khác . Nếu như với các biến thông thường, khi trang web bất kỳ bắt đầu thực thi, biến đó sẽ được cấp phát bộ nhớ, lưu giá trị và thu hồi vùng nhớ sau khi trang kết thúc. Session sẽ khác, nó có thể được tạo ra, tồn tại trên server , có thể xuyên từ trang này sang trang khác, chỉ mất đi khi ta xóa nó hoặc hết tuổi thọ (quá thời gian load dữ liệu hoặc thoát khỏi địa chỉ trang-đóng ứng dụng). |
| 48 | **Phân biệt Session và Cookie**   |  |  | | --- | --- | | **Cookie** | **Session** | | Cookie được lưu trữ trên trình duyệt của người dùng. | Số phiên không được lưu trữ trong trình duyệt của người dùng. | | Dữ liệu cookie được lưu trữ ở phía máy khách. | Dữ liệu session được lưu trữ ở phía máy chủ. | | Dữ liệu cookie dễ dàng sửa đổi khi chúng được lưu trữ ở phía khách hàng. | Dữ liệu phiên không dễ dàng sửa đổi vì chúng được lưu trữ ở phía máy chủ. | | Dữ liệu cookie có sẵn trong trình duyệt của chúng ta đến khi hết hạn. | Dữ liệu phiên có sẵn cho trình duyệt chạy. Sau khi đóng trình duyệt sẽ mất thông tin phiên. | |
| 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**  Web service (Dịch vụ web) là các thành phần ứng dụng được hiển thị dưới dạng các dịch vụ trên WWW. • Dịch vụ • Là một thành phần phần mền • Chứa một vàu logic nghiệp vụ bên trong đó • Được hiển thị trên Web cho nhiều loại client • Được truy cập bởi các client tại các vị trí khác nhau • Web Service có thể sử dụng để tích hợp với các ứng dụng được viết bằng các ngôn ngữ khác nhau và chạy trên các nền tảng khác nhau. • Xây dựng các chuẩn mở và sử dụng các giao thức mở để giao tiếp |
| 51 | **Phân biệt Web Service và Web truyền thông**   |  |  | | --- | --- | | Website | Web Service | | Có giao diện người dùng hoặc GUI | Không có giao diện người dùng | | Nói đến Website được hiểu sử dụng bởi con người | Nói đến Web Service được hiểu sử dụng bởi các ứng dụng được tương tác với nhau qua internet | | Website hoạt động đa nền tảng, vì chúng yêu cầu tinh chỉnh để hoạt động trên các trình duyệt hay hệ điều hành khác nhau. | Web Service độc lập về nền tảng và tất cả dạng truyền thông đều sử dụng giao thức chuẩn. | | Website được truy cập bởi các thành phần trong giao diện người dùng như button, textbox, form ... | Webservice được truy cập bởi phương thức HTTP– PUT, GET, POST, DELETE ... | |
| 52 | **REST là gì? RESTful là gì?**  **REST** (Representational State Transfer) được sử dụng để tạo ra và giao tiếp với Web Service thay thế mô hình SOAP. • REST có kiến trúc đón giản, định rõ các ràng buộc nhằm tạo ra ứng dụng Web Service đạt được những tính chất mong muốn về hiệu suất, khả năng mở rộng, khả năng điều chỉnh v.v.. • REST hướng tới việc xây dựng ứng dụng Web Service có khả năng làm việc tốt nhất trên môi trường WWW. • Dữ liệu và các tính năng được coi như tài nguyên và được truy xuất thông qua các URI (Uniform Resource Identifier) • Sử dụng 4 phương thức chính của HTTP là POST, GET, PUT và DELETE để thực hiện các hành động CRUD đối với cá tài nguyên.  **RESTfu**l web service là một web service được xây dựng dựa trên cấu trúc REST. • RESTful web service tuân thủ các quy tắc cơ bản của cấu trúc REST gồm: • Sử dụng các phương thức HTTP một cách rõ ràng • Phi trạng thái • Hiển thị cấu trúc thư mục như các URls • Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai.  Sử dụng các phương thức HTTP một cách rõ ràng • Để tạo một tài nguyên trên máy chủ, sử dụng phương thức POST. • Để truy xuất một tài nguyên, sử dụng phương thức GET. • Để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó, sử dụng phương thức PUT. • Để huỷ bỏ hoặc xoá một tài nguyên, sử dụng phương thức DELETE |
| 53 | **jQuery là gì? Ajax là gì?**  AJAX – Asynchronous JavaScript and XML, là một nhóm các công nghệ phát triển web được sử dụng để tạo các ứng dụng web giàu tính tương tác, nhanh hơn và mượt mà hơn với sự giúp đỡ của XML, HTML, CSS và JavaScript. • AJAX cho phép các trang web được cập nhật một cách không đồng bộ bằng cách trao đổi các lượng dữ liệu nhỏ với Server mà không cần phải tải lại. • Với AJAX, người dùng có thể tiếp tục sử dụng ứng dụng trong khi các chương trình trên Client tạo Request để lấy thông tin từ Server. |
| 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?**  I**nternationalization** Là quá trình thiết kế một ứng dụng phần mềm để nó có thể đáp ứng được nhiều ngôn ngữ và khu vực khác nhau mà không cần thay đổi kỹ thuật. • Còn được gọi là Translation hay Localization Enablement • Được gọi tắt là i18n, 18 là số chữ cái nằm g  **Localization** • Là quá trình điều chỉnh phần mềm đã được quốc tế hóa cho một ngôn ngữ hoặc một khu vực cụ thể bằng cách chỉ định ngôn ngữ hoặc khu vực sau đó dịch văn bản. • Được gọi tắt là l10n, 10 là số chữ cái nằm giữa L và N • Sự kết hợp giữa i18n và l10n đôi khi được gọi là Globalization và được gọi tắt là g11n • Trong giao tiếp phổ thông, i18n được ngầm hiểu là g11n |
| 57 | Nêu cách triển khai I18N và I10N trong dự án Spring |
| 58 | **Interceptor là gì?**  Được đính vào trong một phần vòng đời của request • Có khả năng xử lý tiền/hậu kỳ các request • Được dùng rất phổ biến để internationalize |
| **Bảo mật** | |
| 59 | **Bảo mật là gì? Cơ chế bảo mật trong Spring như thế nào?**  Cơ chế bảo mật ứng dụng • Xác thực (Authentication) • Phân quyền (Authorization) • Phân quyền hệ thống tệp tin • Quản lý tài khoản • Sử dụng SSL để truyền dữ liệu • Mã hoáFirewalls |
| 60 | **Authentication là gì ? Authorization là gì ?**  • **Authentication** (xác thực) là một hành động nhằm thiết lập hoặc chứng thực một thông điệp hoặc đối tượng nào đó là đáng tin cậy. Điều này có nghĩa là chúng ta có thể tin tưởng vào những lời khai báo hoặc thông điệp mà đối tượng ấy đưa ra.  • **Authorization** (Cấp quyền/phân quyền) là quá trình xác định xem một người dùng có quyền truy cập một tài nguyên cụ thể để thực hiện một số hành động hay không. • Authorization còn được hiểu là sự phân quyền người dùng để họ có quyền truy cập và sử dụng những tính năng mà ứng dụng cung cấp |
| 61 | **Liệt kê các cơ chế xác thực**  HTTP Basic • Cookies • Tockens • Signature • One-time password |
| 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. CSRF là kỹ thuật tấn công vào người dùng, dựa vào đó hacker có thể thực thi những thao tác phải yêu cầu sự chứng thực. 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. |
| **Spring Boot** | |
| 64 | **Spring Boot là gì?**  Spring Boot là giải pháp cho việc cấu hình một project độc lập của Spring. • Spring Boot giúp cho việc bắt đầu một dự án Spring trở lên dễ dàng và nhanh chóng. • Spring Boot được thể hiện thông qua một loạt các quy tắc nhằm mục đích. • Cung cấp ra các ứng dụng đã được cấu hình và đóng gói đầy đủ có thể chạy độc lập. • Giảm thiểu tối đa sự phức tạp của việc cấu hình project cũng như quản lý các thư viện được cung cấp từ bên thứ 3 ngoài Spring |
| 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? |