|  |  |
| --- | --- |
| 1. Trình bày cách Spring MVC xử lý request | 1. Request từ người dùng tới ứng dụng web của chúng ta, **DispatcherServlet**sẽ là đối tượng nhận request. 2. Tìm và điều hướng request tới handler phù hợp, ở đây là các Controllers trong ứng dụng web của chúng ta. 3. Xử lý request 4. Chuẩn bị model và chọn view hiển thị. 5. Trả về kết quả xử lý request cho **DispatcherServlet.** 6. **DispatcherServlet**sẽ gọi View Template phù hợp để xử lý việc hiển thị trên giao diện bằng cách sử dụng model. 7. View template trả kết quả về cho **DispatcherServlet.** 8. Trả response về cho người dùng. |
| 2 .Trình bày cơ chế Dependency Injection trong Spring | Dependency Injection là cơ chế tiêm sự phụ thuộc để giảm sự phụ thuộc.  Cơ chế:  Khi chương trình chạy thì IoC container sẽ thực hiện scan các pagkage để tìm các class được đánh dấu là bean dưới dạng annotation như @Repository, @Controller, @Service… . Khi có một bean phụ thuộc vào bean khác, thì IoC sẽ tìm trong container, nếu chưa có thì tạo, nếu đã có thì lấy ra và tiêm vào bean cần dùng. |
| 3. Có bao nhiêu cách để thực hiện Dependency Injection? Trình bày? | Có 3 cách để thực hiện Dependency Injection.   1. Sử dụng annotation @Autowired: Ta dùng annotation @Autowired để báo cho Spring biết tự động tìm và tiêm bean phù hợp vào vị trí đặt annotation. Trường hợp có 2 class cùng implement 1 interface thì khi muốn tiêm 1 trong 2 class đó chúng ta phải đặt tên cho @Component(“name”) đó và dùng annotation @Qualifier (“name”) hoặc dùng @Primary đánh dấu lên một bean. Khi đó bean này sẽ được ưu tiên chọn trước. 2. Inject qua constructor: Code inject theo kiểu constructor-based nên dùng khi các module là bắt buộc. Khi đó Spring Boot khi tạo bean (cũng chỉ là tạo object, gọi constructor thôi) thì sẽ đưa các phụ thuộc vào constructor khi gọi. 3. Inject qua setter: Cách dùng setter để inject thường dùng trong trường hợp phụ thuộc vòng, module A phụ thuộc vào B và ngược lại. Do đó, nếu cả hai đều sử dụng constructor based injection thì Spring Boot sẽ không biết nên tạo bean nào trước. |
| 1. Framework là gì ? Framework khác Library chỗ nào ? | Library là một tập hợp các functions, class được viết sẵn để có thể tái sử dụng. Mỗi function hay class sẽ thực hiện cho một công việc cụ thể nào đó.  Ví dụ như JQuery là một library, nó cung cấp các function giúp chúng ta thao tác với DOM.  Framework 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.  **Khác nhau:**  – Framework và Library đều cung cấp các tính năng (functions) được viết sẵn để chúng ta có thể tái sử dụng. – Framework lớn hơn và phức tạp hơn Library. – Sử dụng Framework bạn phải thay đổi cấu trúc code của dự án (project’s structure) theo các quy tắc của framework đó để có thể sử dụng được các functions mà framework đó cung cấp. – Chúng ta có thể sử dụng các functions của Library một cách trực tiếp mà không cần thay đổi cấu trúc code của dự án. – Framework có thể hiểu là một khung chương trình, người dùng bổ sung code và tuân theo quy tắc để tạo ra ứng dụng. Còn Library chỉ cung cấp các chức năng tiện ích hay các class để sử dụng trong quá trình xây dựng ứng dụng. – Framework hoạt động chủ động. Nghĩa là nó có thể đưa ra các quyết định gọi hoặc bị gọi bởi các Library hay ứng dụng nào đó. – Library hoạt động bị động. Nghĩa là nó chỉ được gọi khi nào chúng ta cần dùng nó. |
| 1. Spring Framework là gì ? | Spring Framework là một trong những khung ứng dụng dựa trên Java phổ biến nhất. Spring Framework là một khung mã nguồn mở có thể được sử dụng để phát triển các ứng dụng Java một cách dễ dàng và với tốc độ nhanh chóng. Nó là một khung công tác nhẹ cũng cung cấp hỗ trợ cơ sở hạ tầng được xác định rõ ràng để phát triển các ứng dụng trong Java. Nói cách khác, bạn có thể nói rằng Spring xử lý cơ sở hạ tầng để bạn có thể tập trung hơn vào việc phát triển ứng dụng của mình. |
| 1. Lợi ích của Spring Framework? | **1) Đảo ngược kiểm soát (IoC)**  Spring cung cấp tính năng chèn phụ thuộc hoặc Inversion of control trong đó lập trình viên không cần tạo đối tượng thay vào đó họ chỉ cần mô tả rằng nó sẽ được tạo như thế nào? Hơn nữa, các nhà phát triển hoặc lập trình viên không cần phải kết nối trực tiếp các thành phần ứng dụng và dịch vụ của chương trình với nhau. Ở đây, người lập trình chỉ phải mô tả rằng thành phần nào sẽ sử dụng dịch vụ nào trong tệp XML. Tất cả các thành phần chỉ bị ràng buộc bởi khung công tác Spring.  **2) Trọng lượng nhẹ**  Nếu chúng ta nói về độ trong suốt và kích thước thì Spring là một framework nhẹ. Khung cơ bản Spring chỉ có kích thước khoảng 1MB và chi phí xử lý của Spring không quá nhiều.  **3) Định hướng theo khía cạnh**  Lập trình hướng khía cạnh cũng được hỗ trợ bởi Spring. Trong mô hình lập trình hướng theo khía cạnh, các chức năng phụ hoặc chức năng hỗ trợ được tách biệt khỏi logic nghiệp vụ chính của chương trình chính. Mối quan tâm cắt ngang có thể được tách biệt trong công nghệ AOP mà khó có thể làm được trong lập trình hướng đối tượng. Trong AOP, mô-đun ứng dụng được tăng lên và việc bảo trì nó trở nên khá dễ dàng hơn.  **4) MVC Framework**  Spring framework đi kèm với ứng dụng web MVC được xây dựng trên các chức năng chính của Spring. Khung công tác có thể cấu hình cao thông qua các công nghệ nhiều chế độ xem như JSP, iText, Tiles, POI và Velocity, v.v. Các lập trình viên cũng có thể sử dụng các khung công tác khác cùng với khung công tác Spring. 5) Dụng cụ chứa Cấu hình và vòng đời của các đối tượng của ứng dụng được quản lý bởi Spring framework. Chúng không chỉ được quản lý bởi Spring mà còn được chứa trong khuôn khổ này. 6) Xử lý ngoại lệ JDBC Spring có một lớp trừu tượng JDBC cụ thể và nó cung cấp một hệ thống phân cấp ngoại lệ cũng có ý nghĩa và đơn giản hóa chiến lược xử lý lỗi. Vì nó có thể được tích hợp với JDO, Hibernate và iBATIS, do đó cũng cung cấp các dịch vụ tích hợp tốt nhất. |
| 1. 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.  Hình bên là Class A ở hai trường hợp, áp dụng IoC và không. Lớp này nó làm việc phụ thuộc vào hai lớp Class B và Class C (B và C gọi là các dependency)  Ở mô hình không IoC, Class A khi cần chủ động tạo ra đối tượng lớp Class B và Class C (nó nắm quyền khởi tạo, điều khiển)  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.Bean là gì? | Spring Beans chính là những Java Object mà từ đó tạo nên khung sườn của một Spring application.Chúng được cài đặt, lắp ráp và quản lý bởi Spring IoC container. Những bean này được tạo ra bởi configuration metadata được cung cấp từ container, ví dụ, trong tag nằm trong file XML.  Các bean được define trong spring framework là singleton bean. Có một thuộc trính trong bean với tên là “singleton” nếu được gán giá trị là true thì bean đó sẽ trở thành singleton, nếu là false thì bean đó sẽ trở thành prototype bean. Mặc định nếu không được định nghĩa giá trị của nó sẽ là true. Vì thế tất cả các bean trong spring framework mặc định sẽ là singleton bean. |
| 9. Trong Spring có bao nhiêu Bean Scope? | Có 5 scope được định nghĩa cho Spring Bean:   * **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. |
| 10.@Autowire là gì? | @Autowired đánh dấu cho Spring biết rằng sẽ tự động inject bean tương ứng vào vị trí được đánh dấu. |
| 11.@Component có ý nghĩa gì? | @Component là một Annotation (chú thích) đánh dấu trên các Class để giúp Spring biết nó là một Bean. |
| 12.Trình bày ý nghĩa của Controller | Trong mô hình MVC, 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. |
| 13.Trình bày ý nghĩa của ModelAndView Class | Là sự kết hợp của 2 khía cạnh truyền dữ liệu và view. |
| 14.Trình bày ý nghĩa của ModelMap | 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. |
| 15.Trình bày ý nghĩa của ViewResolver Interface | bước cuối cùng của việc xử lý request, DispatcherServlet sẽ chọn View Template và truyền Model data để View Template có thể xử lý, render giao diện và trả về lại cho DispatcherServlet. View Template sẽ là những đối tượng hiện thực interface ViewResolver và chúng sẽ quyết định việc render giao diện sẽ xảy ra như thế nào dựa trên tên view mà chúng ta trả về trong controller. Spring MVC có nhiều đối tượng hiện thực interface ViewResolver, một trong số đó chúng ta có InternalResourceViewResolver. |
| 16.Phân biệt POST với PUT thường sử dụng để làm gì? | POST để tạo dữ liệu còn PUT thì để cập nhật lại dữ liệu.  PUT 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. |
| 17.@RequestMapping làm gì? | @RequestMapping là một trong những annotation phổ biến nhất được sử dụng trong các ứng dụng Web Spring. Annotation này ánh xạ các HTTP request tới các phương thức xử lý của MVC và REST controller. |
| 18.Thuộc tính consumes trong các Request Mapping là gì ? | Để quy định định dạng dữ liệu mà một request có thể process khi người dùng request tới, ví dụ như định dạng dữ liệu JSON, chúng ta có thể sử dụng annotation @RequestMapping với thuộc tính consumes trong Spring MVC.  Đọc thêm: https://huongdanjava.com/vi/su-dung-annotation-requestmapping-voi-thuoc-tinh-consumes-trong-spring-mvc.html |
| 19. Thuộc tính produces trong các Request Mapping là gì ? | Để quy định định dạng dữ liệu sẽ trả về cho người dùng khi họ request một URL nào đó, trong Spring MVC chúng ta có thể sử dụng annotation @RequestMapping với thuộc tính produces.  Đọc them: https://huongdanjava.com/vi/su-dung-annotation-requestmapping-voi-thuoc-tinh-produces-trong-spring-mvc.html |
| 20. Trình bày cơ chế Data Binding trong Spring | 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.  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 (dữ liệu của bean đặt trong model). 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.  Một cách hiểu khác là ràng buộc dữ liệu có thể là một chiều hoặc hai chiều   * Chiều đi: Chuyển dữ liệu từ các điều khiển trên form vào các thuộc tính của đối tượng dữ liệu (bean) * Chiều về: Hiển thị dữ liệu từ của các thuộc tính của đối tượng lên các điều khiển của form |
| So sánh RequestParam và PathVariable ? | * @RequestParam được sử dụng để truy cập (lấy) giá trị của parameters trên URL(kiểu query string) * @PathVariable được sử dụng để lấy giá trị trên URI theo template (còn gọi là URI template). |
| 22. Thymeleaf là gì? | Thymeleaf là một Java template engine dùng để xử lý và tạo HTML, XML, Javascript, CSS và text.  Mục tiêu chính của thymeleaf là mang lại các template tự nhiên, đồng nhất, đơn giản (nature templates) cho công việc phát triển.  Các lợi ích của Thymeleaf Với thymeleaf, ta chỉ cần sử dụng file HTML là có thể hiển thị tất cả mọi thứ (không cần jsp ...). Thymealeaf sẽ tham gia vào renderd các file HTML dưới dạng các thuộc tính trong các thẻ HTML --> do đó ta không cần phải thêm bất kỳ thẻ non-HTML nào. Vì là HTML nên ta có thể xem các file view mà không cần khởi chạy server. Thymeleaf hỗ trợ cơ chế cache, do đó ta có thể cache dữ liệu hoặc custom để hiển thị view khi có thay đổi mà không cần restart server. |
| 45. Phân biệt Session & Cookie | |  |  | | --- | --- | | **Cookie** | **Session** | | **Cookie** được lưu trữ trên trình duyệt của người dùng | **Session** không được lưu trữ trong trình duyệt 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 **Session** 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 **Session** có sẵn cho trình duyệt chạy. Sau khi đóng trình duyệt sẽ mất thông tin **Session** | | **Lưu** ý: Khi truy cập trang web, Server sẽ tạo ngẫu nhiên 1 ID định danh cho phiên hoạt động của người dùng và đồng thời máy của người dùng cũng tự sinh 1 Cookie để lưu trữ giá trị Session ID đó. Việc này giúp server quản lý được các request gửi lên để có thể trả response về đúng client đã request | | |
| 46. Session là gì? Cho ví dụ về Session? | - Sesion là một phiên làm việc được dùng trong lập trình web và có thể kết nối database  VD: Thực hiện chức năng đăng nhập, đăng xuất |
| 47. Cách triển khai Session | B1: Sử dụng @SessionAttributes để khai báo Session  B2: Sử dụng @ModelAttribute bên ngoài handle method để khởi tạo Session  B3: Sử dụng @ModelAttribute trong tham số của handle method của Controller tạo ra nó để tương tác với Session  B4: Sử dụng @SessionAttribute trong tham số của handle method của Controller không tạo ra Session đó để tương tác với Session. |
| 48. Web Service là gì? Lấy ví dụ về web service | - Web service là tập hợp các giao thức và tiêu chuẩn mở được sử dụng để trao đổi dữ liệu giữa các ứng dụng hoặc giữa các hệ thống. Tóm gọn:  + Là phương thức giao tiếp giữa hai thiết bị qua mạng  + Là ứng dụng hoặc thành phần ứng dụng để giao tiếp  + Là tập hợp các tiêu chuẩn hoặc giao thức để trao đổi thông tin giữa 2 thiết bị hoặc ứng dụng  - Các ứng dụng phần mềm được viết bằng các ngôn ngữ lập trình khác nhau và chạy trên nhiều nền tảng khác nhau, có thể sử dụng web service để trao đổi dữ liệu qua mạng máy tính  - Web service hoạt động một cách độc lập không phụ thuộc bất kì ngôn ngữ nào. Các ứng dụng java, .net hoặc PHP… có thể giao tiếp với các ứng dụng khác thông qua web service  VD: Chúng ta ứng dụng có FE phát triển từ Angular thực hiện việc tiếp nhận request. Sau đó gởi dữ liệu về Web service, web service sẽ gởi request về BE được viểt bằng Spring. BE xử lý request và response về cho web service, web service gởi lại response về FE và FE xử lý dữ liệu và trả về dữ liệu mà người dùng có thể hiểu được. |
| 49. Phân biệt Web Service và Web truyền thống | |  |  | | --- | --- | | **Website** | **Web service** | | - Có giao diện | - Không có giao diện | | - Được sử dụng bởi con người | - Được sử dụng bởi các ứng dụng khác | | - Website hoạt động đa nền tảng vì chúng cần tinh chỉnh để phù hợp các trình duyệt hay hệ điều hanh khác nhau | - Web service độc lập về nền tảng, sử dụng giao thức chuẩn để giao tiếp | | - Website được truy cập thong các thành phần của giao diện như button, form | - Truy cập thông qua các phương thức HTTP: get, post, put, delete, patch | | - Website là ứng dụng đầu cuối, người dùng truy cập website qua URL, qua đó nhận được những dữ liệu như text, hình ảnh, âm thanh có thể dễ dàng hiểu được | - Là 1 khái niệm rộng hơn, dữ liệu trả về từ web service, người dùng khó có thể hiểu được. Dữ liệu đó được các ứng dụng khác sử dụng và chế biến thành các dữ liệu mà người dùng có thể đọc được | |
| 50. SOAP là gì? RESTful là gì? | |  |  | | --- | --- | | **SOAP** | **REST** | | - SOAP (Simple Object Access Protocol – Giao thức truy cập đối tượng đơn giản) là giao thức nhắn tin cho phép các chương trình chạy trên các hệ điều hành khác nhau (như Window hay Linus) giao tiếp được với nhau thông qua giao thức HTTP và ngôn ngữ XML | - REST (Representational State Transfer) là một kiểu kiến trúc không phải là một giao thức. Cho phép định dạng nhiều kiểu dữ liệu khác nhau như: Plain Text, HTML, XML và JSON | | - Ưu điểm:   + SOAP định nghĩa bảo mật riêng cho nó được gọi là WS Security  + Có thể được viết bằng bất kỳ ngôn ngữ lập trình nào và được thực thi trong bất kỳ nền tảng nào | - Ưu điểm:  + Nhanh: RESTful nhanh hơn vì không đặc tả nghiêm ngặt như SOAP. Chiếm ít băng thông và tài nguyên hơn  + Có thể được viết bằng bất kỳ ngôn ngữ lập trình nào và được thực thi trong bất kỳ nền tảng nào | | - Nhược điểm:  + Chậm: SOAP sử dụng định dạng XML phải được phân tích cú pháp. Các ứng dụng SOAP phải tuân theo nhiều tiêu chuẩn. Vì vậy nó chậm và chiếm nhiều băng thông và tài nguyên |  | |
| 51. jQuery là gì? Ajax là gì? | |  |  | | --- | --- | | **jQuery** | **Ajax** | | - jQuery là thư viện được viết từ JS, jQuery giúp xây dựng các chức năng bằng JS dễ dàng, nhanh và giàu tính năng hơn | - Asynchronous JS and XML là nhóm các công nghệ phát triển web được sử dụng để tạo các trang web giàu tính tương tác, nhanh hơn, mượt mà hơn với sự giúp đỡ của XML, HTML, JS, CSS thông qua cơ chế bất động bộ | |
| 52. Ví dụ về cách triển khai Ajax | $.ajax(url, option) - Các tham số của option:  + type, url, contentType, success, errror |
| 53. @RequestHeader và @ResponseHeader có ý nghĩa gì | |  |  | | --- | --- | | **@RequestHeader** | **@ResponseHeader** | | - Được sử dụng khi muốn lấy dữ liệu được truyền bằng Header của một request (yêu cầu từ client) | - Được sử dụng khi muốn trả về thêm dữ liệu cho client ở phần trên cùng của mỗi response | |
| 57. Bảo mật là gì? Cơ chế bảo mật trong Spring như thế nào? | - Bảo mật là sự hạn chế khả năng lạm dụng tài nguyên và tài sản.  - Cơ chế bảo mật trong Spring hoạt động dựa trên Authentication và Authorization |
| 58. Authentication là gì ? Authorization là gì ? | |  |  | | --- | --- | | **Authentication** | **Authorization** | | - Authentication là hoạt động nhằm thiết lập hoặc chứng thực một thông điệp hoặc một đối tượng nào đó đáng tin cậy | - Authorization / Access Control là tiến trình quyết định xem 1 principal có được phép thực hiện một hành động nào đó trong ứng dụng hay không | |
| 59. Các cơ chế xác thực ? | - OTP (One Time Password): mật khẩu sử dụng một lần, thường được sử dụng như lớp bảo mật cuối cùng trong các giao dịch trực tuyến, đặc biệt là ngân hàng điện tử.  - Oauth2: là phương thức xác thực giúp một ứng dụng bên thứ 3 có thể được ủy quyền bởi người dùng để truy cập đến taì nguyên người dùng nằm trên một dịch vụ khác  - JWT (JSON Web Token): là 1 chuẩn mở (RFC 7519) có thể giúp tạo ra một chuỗi mã hóa chứa các dữ liệu dùng để trao đổi thông tin giữa các hệ thống khác nhau một các an toàn và đáng tin cậy. Bao gồm:  + Header (tiêu đề): chứa loại mã thông báo hoặc thuật toán băm  + Payload (tải trọng) : chứa các thông tin người dùng  + Signature (chữ ký): được tạo ra bằng cách mã hóa phần header và payload kèm theo một chuỗi secret (khóa bí mật) |
| 60. CSRF là gì | - CSRF (Cross-site Request Forgery) là kỹ thuật tấn công giả mạo chính chủ thể của nó bằng cách sử dụng quyền chứng thực của người dùng đối với 1 website |
| 61. CORS là gì | - CORS (Cross-origin Resource Sharing) là một cơ chế cho phép nhiều tài nguyên khác nhau (fonts, JS, …) của một trang web có thể được truy vấn từ domain khác với domain của trang đó. |
| 62. Spring Boot là gì? | - Spring Boot là một trong số những module của Spring Framework chuyên cung cấp các tính năng **Rapid Application Development (RAD)** để tạo ra và phát triển các ứng dụng độc lập dựa trên Spring. |
| 63. Sự khác nhau giữa SpringBoot và SpringMVC là gì? | |  |  | | --- | --- | | **Spring MVC** | **Spring Boot** | | - 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 có thể 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 trẻn 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. Spring tích hợp một loạt các module khác nhau trong ô của nó như: spring-core, spring-data, spring-web (bao gồm Spring MVC)… 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 | | - Spring MVC là một khung được sử dụng trong các ứng dụng web | - Spring Boot là trình khởi tạo dự án dựa Spring sản xuất sẵn sàng | |
| 64. Cấu hình cho Spring Boot. Tìm kiếm các Bean ở nhiều package khác nhau bằng cách nào? | - Đối với các ứng dụng Spring Boot, @SpringBootApplication là sự kết hợp của 3 annotation khác là :  + @Configuration  + @EnableAutoConfiguration  + @ComponentScan  Spring Boot sẽ quét tất cả các class ở cùng package với main class và tất cả các sub-package của nó để tìm và khởi tạo các bean tương ứng đã được định nghĩa  - Và đối với Class ở khác package thì cần chỉ định đây là nơi cần tạo bean bằng @ComponentScan để Spring nhận vào danh sách package sẽ quét để khởi tạo bean |
| 65. Trình bày Spring Security? @EnableWebSecurity làm gì? | - Spring Security là một loạt các bộ lọc servlet giúp thêm các authentication và authorization vào ứng dụng web  - Thêm @EnableWebSecurity vào lớp @Configuration để xác định cấu hình Spring Security trong bất kỳ WebSecurityConfigurer nào hoặc nhiều khả năng hơn bằng cách mở rộng lớp cơ sở WebSecurityConfigurerAdapter và ghi đè từng phương thức riêng lẻ |
| 66. Trình bày Spring Boot JPA? Cài đặt? | - Spring Boot JPA giúp sử dụng JPA trong Spring một cách dễ dàng hơn  - Có thể sử dụng với các framework hibernate, OpenJPA, EclipseLink  - Cải tiến JPA tiêu chuẩn, đơn giản hóa tầng truy xuất, tự tạo repository, tự tạo ra các truy vấn thông qua hàm ghi log, phân trang |
|  |  |
|  |  |
|  |  |

1. **Sử dụng lặp trong Thymeleaf như thế nào? Sử dụng điều kiện trong Thymeleaf như thế nào?**

* **th:each = "object, iterStat : S{array}"**
* Đối tượng iterStat cung cấp các trạng thái:
* Index: chỉ số hiện tại, bắt đầu từ 0
* Count: chỉ số, bắt đầu từ 1
* Size: tổng số phần tử
* Current: even, odd, first, last

1. **Formatter là gì ? Converter là gì ?**

* Formatter và converter đều là 2 thành phần hỗ trợ chuyển đổi dữ liệu nhập vào sang kiểu dữ liệu thích hợp.
* Kiểu dữ liệu nguồn của formatter là String, trong khi đó converter có thể làm việc với bất kỳ 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.

**25 ORM là gì ?**

* ORM (**O**bject **R**elational **M**apping), là một kỹ thuật/cơ chế lập trình thực hiện ánh xạ CSDL sang các đối tượng trong các ngôn ngữ lập trình hướng đối tượng như Java, C# …(các table tương ứng các class, mối ràng buộc giữa các table tương ứng quan hệ giữa các class ‘has a’ , ‘is a’).
* Việc sử dụng ORM cho phép lập trình viên thao tác với database 1 cách hoàn toàn tự nhiên, dễ hiểu thông qua các đối tượng. Lập trình viên không cần tới loại database, kiểu dữ liệu trong database…
* **ƯU ĐIỂM ORM:**
* OOP: ORM giúp lập trình viên tập trung vào lập trình hướng đối tượng
* Tính độc lập: Làm việc được với nhiều loại database(hệ quản trị cơ sở dữ liệu), nhiều kiểu dữ liệu khác nhau. Dễ dàng thay đổi loại database hơn. Các câu lệnh SQL không phụ thuộc vào loại database.
* Đơn giản, dễ sử dụng: Hỗ trợ HSQL, cung cấp nhiều nhiều API truy vấn.
* Năng suất hơn: viết code ít hơn, dễ hiểu hơn. Phù hợp các case CRUD (Create, Read, Update, Delete)
* Khả năng sử dụng lại code.
* **NHƯỢC ĐIỂM:**
* Khả năng truy vấn bị hạn chế, nhiều trường hợp ta vẫn phải dùng native SQL để truy vấn database.
* Khó tối ưu câu lệnh SQL (do câu lệnh SQL được ORM tự động sinh ra).

**26 JPA là gì? Spring Data JPA là gì ?**

* JPA (Java Persistence API) là một bản ghi chi tiết của Java để quản lý dữ liệu **quan hệ** trong các ứng dụng Java. Nó cho phép chúng ta truy cập và lưu trữ dữ liệu giữa các object/class Java và database quan hệ. JPA tuân theo **Object-Relation Mapping** (ORM). Nó là một tập hợp các interface. Nó cũng cung cấp một API **EntityManager** runtime để xử lý các câu query và giao dịch trên các object dựa trên database. Nó sử dụng ngôn ngữ truy vấn hướng đối tượng độc lập nền tảng JPQL (Java Persistent Query Language).
* Spring Data JPA là tầng được xây dựng phía trên tầng JPA. Có nghĩa là nó sẽ sử dụng tất cả các chức năng của JPA như entity mapping, entity life-lifecycle, JPA query v.v để xây dựng một số tính năng như no-code repository, khởi tạo query thông qua tên hàm v.v.

**27 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 cơ sở dữ liệu quan hệ.
* Hay nói cách khác, Hibernate chính là cài đặt của JPA (JPA là 1 tập các interface, còn Hibernate implements các interface ấy 1 cách chi tiết).
* Code sử dụng Hibernate là độc lập với hệ quản trị cơ sở dữ liệu, nghĩa là ta không cần thay đổi câu lệnh HQL khi ta chuyển từ hệ quản trị CSDL MySQL sang Oracle, hay các hệ quản trị CSDL khác… Do đó rất dễ để ta thay đổi CSDL quan hệ, đơn giản bằng cách thay đổi thông tin cấu hình hệ quản trị CSDL trong file cấu hình.

//used MySQL

com.mysql.jdbc.Driver

// used Oracle

oracle.jdbc.driver.OracleDriver

**28 Liệt kê một số annotation của hibernate?**

* @Entity được sử dụng để chú thích một class là một Entity.
* @Table cho phép chú thích tên bảng thông qua thuộc tính name (thuộc tính này không bắt buộc).
* @Id được sử dụng để mô tả đây là Id (Identity) của Entity, nó tương đương với cột đó là khóa chính (Primary Key) của table trong database.
* @GeneratedValue được sử dụng để Hibernate tự động tạo ra giá trị và gán vào cho một cột trong trường hợp insert mới một Entity vào database. Nó có thể gắn trên cột Id hoặc một cột nào đó.
* @Column được sử dụng để chỉ định thông tin chi tiết của cột mà một field của entity sẽ được ánh xạ với một column trong database.
* @ManyToOne mô tả một quan hệ  **N-1** (Nhiều – Một), nó thường được sử dụng cùng với @JoinColumn.
* @Transient. Trong một số trường hợp, trong entity class có chứa một field mà field này không tồn tồn tại trong database. Khi đó chúng ta sẽ gặp lỗi “java.sql.SQLSyntaxErrorException: Unknown column ‘additionalPropery’ in ‘field list’”. Để tránh lỗi này, chúng ta có thể sử dụng @Transient để thông báo rằng thuộc tính/ phương thức này không liên quan gì tới một cột nào dưới database. Khi đó, Hibernate sẽ bỏ qua field này.

**29 Entity là gì?**

* Trong JPA – Hibernate, một entity là một [Java class](https://shareprogramming.net/object-va-class-trong-java/) tương ứng với một table trong database, mỗi entity object tương ứng với một dòng trong bảng. JPA yêu cầu một entity class phải cung cấp đầy đủ các metadata để ánh xạ giữa entity class và table như tên cột, kiểu dữ liệu, khoá chính, khoá ngoại, etc.

**30 Persistence Context & Entity Manager là gì?**

**PERSISTENCE CONTEXT**

* Ngoài việc map đối tượng Java đến bản ghi trong CSDL (tức là ORM), thì có một vấn đề mà Hibernate phải care đến đó là quản lý các Entity. Cái ý niệm về “persistence context” chính là giải pháp để giúp Hibernate làm được việc này. **Persistence context** có thể coi là một “môi trường” chứa toàn bộ các đối tượng mà ta tạo ra và lưu vào csdl trong mỗi session.
* Một Session, hay là 1 phiên, là một giao dịch, có phạm vi tùy vào từng ứng dụng. Khi ta làm việc với DB thông qua một Persistence Context, mọi thực-thể sẽ gắn vào context này, mỗi bản ghi trong DB mà ta tương tác sẽ tương ứng với 1 thực thể trong context này.
* Trong Hibernate, PersistenceContext được tạo ra nhờ [*org.hibernate.Session*](http://docs.jboss.org/hibernate/orm/5.2/javadocs/org/hibernate/Session.html) . Với JPA, PersistenceContext được thể hiện thông qua class [*javax.persistence.EntityManager*](http://docs.oracle.com/javaee/7/api/javax/persistence/EntityManager.html). [JPA](https://gpcoder.com/6282-tong-quan-ve-jpa-java-persistence-api/) là bộ đặc tả cho việc lưu dữ liệu vào DB dành cho ngôn ngữ Java, Hibernate sau này đã tuân theo bộ đặc tả đó. Khi đó nếu dùng combo JPA-Hibernate, thì Persistence Context được tạo ra bởi EntityManager interface, thực tế sẽ là một lớp bọc lấy cái Session object ở phía dưới. Nếu ta xài thẳng Session (ko xài EntityManager) thì sẽ có nhiều phương thức cho ta xài hơn, tiện dụng hơn.

ENTITY MANAGER:

* EntityManager là một giao diện (interface) cung cấp các API cho việc tương tác với các Entity như Persist (lưu một đối tượng mới), merge (cập nhật một đối tượng), remove (xóa 1 đối tượng).

**31 Câu lệnh truy vấn động là gì? Câu lệnh truy vấn tĩnh là gì?**

### ****Câu lệnh truy vấn động****

EntityManager cung cấp phương thức createQuery() để tạo các câu lệnh truy vấn động. Ví dụ:

@Override

public List<Product> findAll() {

TypedQuery<Product> query = em.createQuery("select p from Product p", Product.class);

return query.getResultList();

}

### ****Câu lệnh truy vấn tĩnh****

Trong các lớp entity, chúng ta có thể định nghĩa sẵn các câu lệnh truy vấn. Ví dụ, đoạn mã sau đây định nghĩa một câu lệnh truy vấn có tên là findAllProductWithName để tìm các sản phẩm có tên mong muốn.

@Entity

@Table(name = "products")

@NamedQuery(name="findAllProductWithName",

query="SELECT c FROM Product c WHERE c.name LIKE :name")

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

@Column

private String name;

@Column

private float price;

}

Sau đó, chúng ta sử dụng phương thức createNamedQuery của EntityManager để tạo câu truy vấn. Ví dụ:

@PersistenceContext

private EntityManager em;

@Override

public List<Product> findByName(String name) {

Query query = em.createNamedQuery("findAllProductWithName", Product.class);

query.setParameter("name", name);

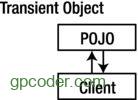
return query.getResultList();

}

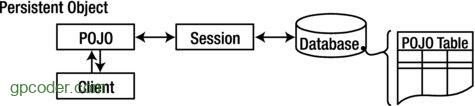
32 Trạng thái của Entity bao gồm những gì?

Một đối tượng trong Hibernate có 1 trong 4 trạng thái:

* **Transient** (Tạm thời): Đối tượng không có quan hệ với Session hiện tại của Hibernate. Đối tượng ở trạng thái này chưa từng gắn vào context, nó không có bản ghi tương ứng trong CSDL

[](https://gpcoder.com/wp-content/uploads/2020/03/Transient-objects-are-independent-of-Hibernate.jpg)

* **Persistent** (Bền vững): Đối tượng đang liên hệ với một context, tức là với một đối tượng **Session** và trạng thái của nó được đồng bộ với cơ sở dữ liệu khi mà ta commit cái Session.

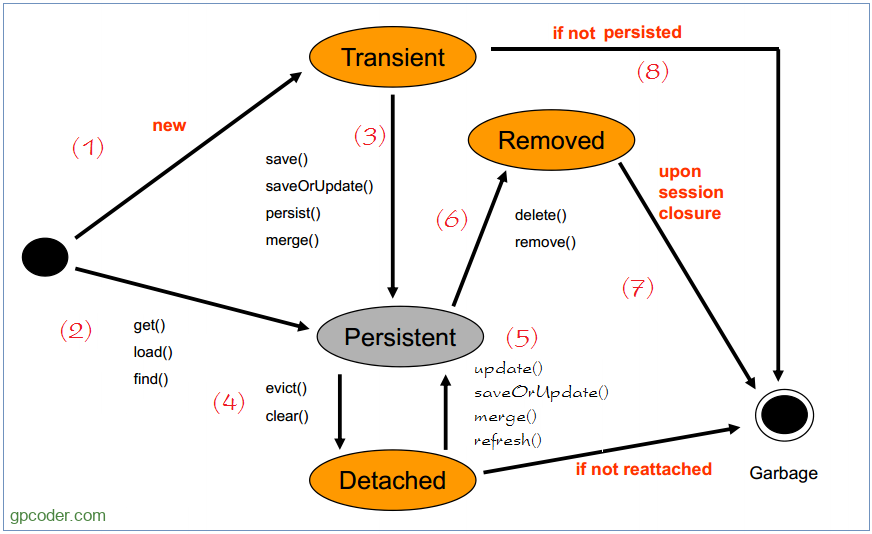
[](https://gpcoder.com/wp-content/uploads/2020/03/Persistent-objects-are-maintained-by-Hibernate.jpg)

* **Detached** (Đã bị tách riêng ra): Đối tượng đã từng có trạng thái persistent nhưng hiện tại đã không còn giữ quan hệ với Session. Nếu nó không được attached trở lại, nó sẽ bị bộ gom rác của Java quét đi theo cơ chế thông thường. Một đối tượng đang trong session muốn đạt đươc trạng thái này thì có những cách là gọi hàm evict(), close Session hoặc làm combo thao tác: serialize/deserialize.

[](https://gpcoder.com/wp-content/uploads/2020/03/Detached-objects-exist-in-the-database-but-are-not-maintained-by-Hibernate.jpg)

* **Removed** (Đã bị xóa): tương tự như detached nhưng bản ghi tương ứng với đối tượng này trước đó đã bị xóa khỏi database.

Sơ đồ bên dưới minh hoạ các trạng thái này:

[](https://gpcoder.com/wp-content/uploads/2020/03/hibernate-lifecycle.png)

* (1) **Transient**: Trường hợp bạn tạo mới một đối tượng java từ một Entity, đối tượng đó có tình trạng là **Transient**. Hibernate không biết về sự tồn tại của nó. Nó nằm ngoài sự quản lý của Hibernate.
* (2) **Persistent**: Trường hợp bạn lấy ra đối tượng Entity bằng method get, load hoặc find, bạn có được một đối tượng nó tương ứng với 1 record dưới database. Đối tượng này có trạng thái **Persistent**. Nó được quản lý bởi Hibernate. Khi đối tượng ở trạng thái persistent, tất cả các thay đổi mà bạn thực hiện đối với đối tượng này sẽ được áp dụng cho các bản ghi và các trường cơ sở dữ liệu tương ứng khi flush session.
* (3) **Transient -> Persistent**: Session gọi một trong các method save, saveOrUpdate, persist, merge sẽ đẩy đối tượng **Transient** vào sự quản lý của Hibernate và đối tượng này chuyển sang trạng thái **Persistent**. Tùy tình huống nó sẽ insert hoặc update dữ liệu vào DB.
* (4) **Persistent -> Detached**: Session gọi evict(..) hoặc clear() để đuổi các đối tượng có trạng thái persistent (bền vững) ra khỏi sự quản lý của Hibernate, giờ các đối tượng này sẽ có trạng thái mới là Detached (Bị tách ra).  Nếu nó không được đính (Attached) trở lại, nó sẽ bị bộ gom rác của Java quét đi theo cơ chế thông thường.
* (5) **Detached -> Persistent**: Sử dụng update(..), saveOrUpdate(..), merge(..) sẽ đính trở lại các đối tượng Detached vào lại. Tùy tình huống nó sẽ tạo ra dưới DB câu lệnh update hoặc insert. Các đối tượng sẽ trở về trạng thái Persistent (bền vững).
* (6) **Persistent -> Removed**: Session gọi method remove(..), delete(..) để xóa một bản ghi, đối tượng persistent giờ chuyển sang trạng thái Removed (Đã bị xóa).

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

**-** Collection mapping

**-** Association mapping

* 4 loại:
* One to one
* One to many
* Many to many
* Many to one

- Component mapping

**34 Để sử dụng interface JpaRepository cần cung cấp những thông tin nào?**

* Cần khai báo kiểu dữ liệu trong Repository triển khai **JpaRepository, ví dụ** public interface ICustomerRepository extends JpaRepository<Customer, Integer>
* Customer: đối tượng cần kết nối database
* Integer: kiểu dữ liệu của Id (khóa chính) trong entity customer

**35 Validation dữ liệu là gì?**

* **Validation**là hành động kiểm tra tính hợp lệ của dữ liệu. Việc này giúp cho hệ thống hoạt động an toàn, đảm bảo, tránh xử lý dữ liệu lỗi,… và cũng giúp ngăn chặn một số cuộc tấn công phổ biến như SQL injection hoặc XSS.
* Ví dụ khi nhập dữ liệu vào form đăng kí, cần thực hiện validate dữ liệu form để đảm bảo:
* Trường username không được trống
* Password phải đủ độ khó
* Email phải đúng định dạng email
* …

**36 Trình bày cách triển khai validate dữ liệu trong Spring – có 2 cách:**

* 1. Validate bằng các annotation:
* @NotNull – kiểm tra giá trị null
* @AssertTrue – kiểm tra giá trị thuộc tính là true
* @Size – kiểm tra độ dài min and max
* @Min – kiểm tra giá trị nhỏ nhất
* @Max – Kiểm tra giá trị lớn nhất
* @Email – kiểm tra email có hợp lệ
* @NotEmpty – kiểm tra không được trống và empty
* @NotBlank – kiểm tra giá trị không được null hoặc khoảng trắng
* @Positive and @PositiveOrZero – kiểm tra chỉ được phép là số nguyên dương từ 0 trở đi
* @Negative and @NegativeOrZero – kiểm tra số âm
* @Past and @PastOrPresent – kiểm tra ngày từ quá khứ đến hiện tại.
* @Future and @FutureOrPresent – kiểm tra ngày từ hiện tại đến tương lai
* 2. Implements interface Validator. Interface này có 2 phương thức là **public boolean supports** và **public void validate,** chúng ta sẽ triển khai thực hiện validate các trường của class trong phương thức **public void validate.**

**37 Binding Result là gì?**

* [BindingResult] là đối tượng của Spring, nơi **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.

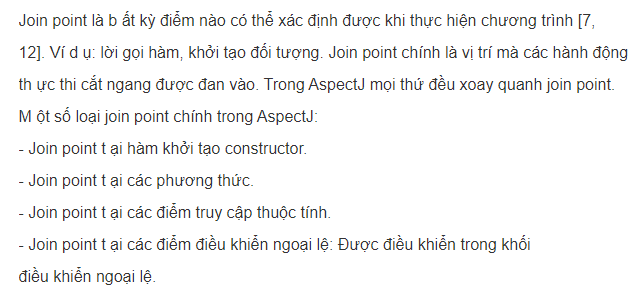
**38 AOP là gì? AOP gồm những yếu tố nào?**

* AOP (Aspect Oriented Programming) là kỹ thuật lập trình hướng khía cạnh, phân tách chương trình thành các module riêng rẽ, phân biệt và không phụ thuộc lẫn nhau.
* AOP không thể thay thế cho OOP mà chỉ hỗ trợ phân tách chương trình thành các module nhỏ để dễ quản lý.
* AOP gồm những yếu tố:
* 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.

**39 Các loại Advice?**

* **Trong Spring AOP, có 4 loại advice được hỗ trợ:**
* Before **advice**: chạy trước khi method được thực thi.
* After returning **advice**: Chạy sau khi method trả về một kết quả
* After throwing adivce: Chạy khi method ném ra một exception.
* Around **advice**: Chạy khi method được thực thi (Bao gồm cả 3 **loại advice** trên)

40 JoinPoint là gì?



41 Nêu cách triển khai AOP mà bạn dùng trong dự án

* Chia nhỏ chương trình thành nhiều module riêng rẽ, hoạt động độc lập và không phụ thuộc nhau
* Khi hoạt động, chương trình sẽ kết hợp các module lại để thực hiện các chức năng nhưng khi sửa đổi 1 chức năng thì chỉ cần sửa 1 module.

@Aspect

@Configuration

public class TestServiceAspect {

private Logger logger = LoggerFactory.getLogger(TestServiceAspect.class);

@Before("execution(\* com.ldt.demospringaop.dao.\*.\*(..))")

public void before(JoinPoint joinPoint){

logger.info(" before called " + joinPoint.toString());

}

* }
* @Before("execution(\* com.ldt.demospringaop.dao.TestDAO.callDaoSuccess(..))")
* public void before(JoinPoint joinPoint){
* logger.info(" before called " + joinPoint.toString());
* }

42 Mục đích xử lý ngoại lệ trong Spring Web MVC ? Cái này anh tìm hiểu và viết ra chứ chưa rõ có đúng với yêu cầu câu hỏi không:

* Khi xảy ra một exception thì hệ thống sẽ báo lỗi, và để che giấu lỗi hệ thống thì chúng ta cần xử lý exception đó và trả về cho client một thông điệp được thể hệ ở phần giao diện web. Ví dụ như khi người dùng nhập chữ vào một ô input cần nhập số thì sẽ xảy ra NumberFormatException, chúng ta bắt lỗi này và gửi về cho client một thông báo ví dụ như alert ra màn hình cho client biết: “cần nhập số”

43 Các cách xử lý ngoại lệ trong Spring Web MVC ?

* Dùng try, catch
* Dùng throw, throws
* Dùng ExceptionHandler

44 Cookie là gì?

* **Http Cookies**, thường được gọi là **cookies (bánh)**. Từ thuở ban đầu dự định sử dụng vào việc lưu trữ thông tin session ở phía client. Theo bản chất, cookies sẽ được gắn với một tên miền (domain) cụ thể.
* Tệp tin cookie sẽ được truyền từ server tới browser và được lưu trữ trên máy tính của client khi client truy cập vào ứng dụng. Như vậy dù có tắt browser cũng không mất đi các giá trị vì chúng ta đã lưu nó trên máy tính của mình.