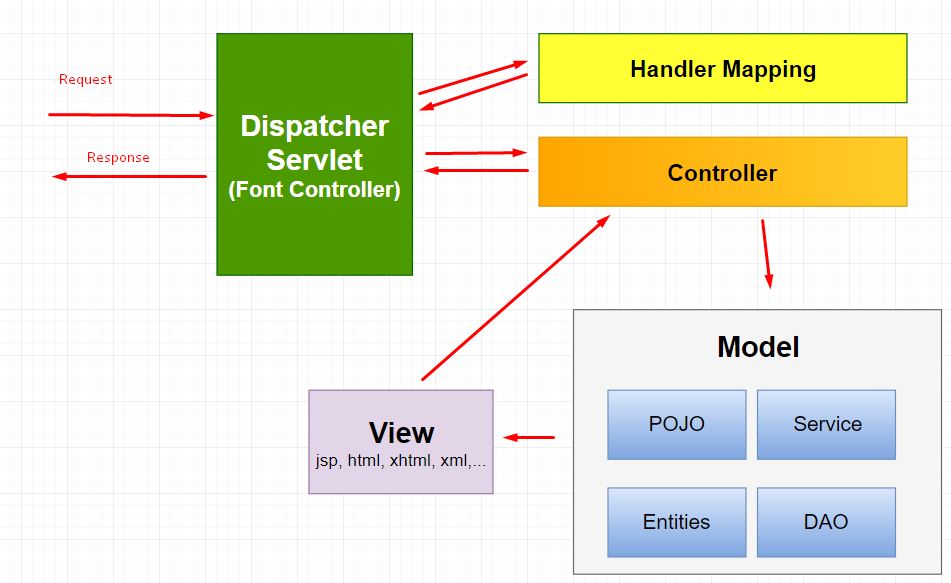
***1. Vẽ và trình bày cách Spring MVC xử lý request***

- Spring MVC Framework là một nền tảng mã nguồn mở phổ biến để phát triển ứng dụng Java EE (ứng dụng web)

- Cung cấp một Front Controller để xử lý hoặc lắng nghe mỗi khi có request tới ứng dụng; điều hướng đến các controller con tương ứng.

- Front Controller chính là Dispatcher Servlet

- DispacherServlet gửi các request tới các Controller và quyết định hồi đáp bằng cách gửi lại view

****

***2. Trình bày cơ chế Dependency Injection trong Spring***

**DI** *(Dependency Injection – tiêm phụ thuộc)* :

* Là một cơ chế để triển khai IoC.
* Là kỹ thuật giảm sự phụ thuộc của 2 module với nhau bằng cách gọi qua Interface
* Triển khai DI bằng cách tiêm đối tượng của module A vào module B.

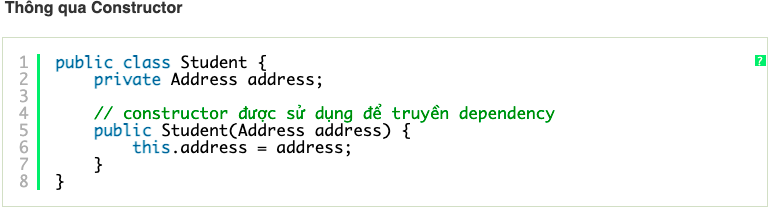
Ví dụ: Controller và ServiveImpl giao tiếp qua Interface Service.

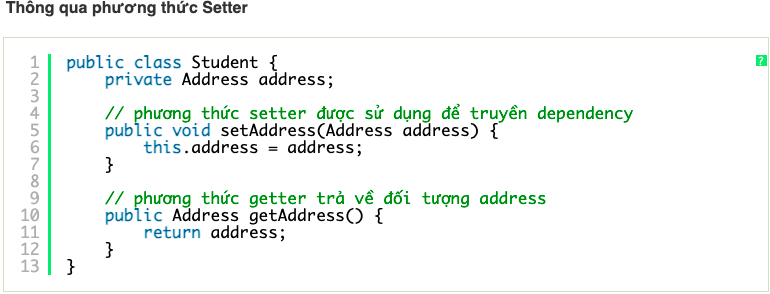
***3. Có bao nhiêu cách để thực hiện Dependency Injection? Trình bày?***

- Có 3 cách triển khai DI:

* Sử dụng Interface
* Sử dụng Contructor
* Sử dụng Setter

Về Interface chắc ai cũng biết rồi còn Setter và Constructor thì nhìn ví dụ dưới sẽ hiểu:





***4. Framework là gì ? Framework khác Library chỗ nào ?***

**- Giống:** cả Framework và Library đều cung cấp các chức năng được viết sẵn để có thể tái sử dụng.

**- Khác***: Điểm khác biệt lớn nhất giữa Framework và Library là cơ chế IoC (Inversion Of Control - đảo ngược quyền điểu khiển)*

* Libs chỉ cung cấp công cụ tiện ích để thoải mái sử dụng. Là ứng dụng nắm quyền điều khiển (control)
* Framework tạo ra một bộ khung và cung cấp các hướng dẫn cần tuân thủ theo. Là framework nắm quyền điều khiển.

***5. Spring Framework là gì ?***

- Spring là một Framework phát triển các ứng dụng Java được sử dụng bởi hàng triệu lập trình viên. Nó giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, sử dụng lại code…Spring là một mã nguồn mở, được phát triển, chia sẻ và có cộng đồng người dùng rất lơn.

- Spring Framework được xây dựng dựa trên 2 nguyên tắc design chính là: Dependency Injection và Aspect Oriented Programming. Những tính năng core (cốt lõi) của Spring có thể được sử dụng để phát triển Java Desktop, ứng dụng mobile, Java Web. Mục tiêu chính của Spring là giúp phát triển các ứng dụng J2EE một cách dễ dàng hơn dựa trên mô hình sử dụng POJO (Plain Old Java Object)

***6. Lợi ích của Spring Framework?***

* Giúp LTV tiết kiệm được tối đa thời gian, công sức khi phát triển phần mềm/ứng dụng.
* Cho phép sản phẩm ứng dụng kế thừa các tính năng, cấu trúc đã được tiêu chuẩn hóa => giúp cho quá trình vận hành và bảo trì/khắc phục sự cố ứng dụng dễ dàng hơn.
* Cho phép người dùng mở rộng tùy ý dựa trên những gì mà **Framework** đã cung cấp.

***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
* **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: Strategy design pattern, Service Locator pattern, Factory pattern, and Dependency Injection (DI)

***8. 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. Các bean này được tạo ra bằng siêu dữ liệu cấu hình mà bạn cung cấp cho container, ví dụ dưới dạng định nghĩa XML <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.

Trong 5 scope trên thì 3 scope cuối chỉ dùng trong ứng dụng web. Bạn cũng có thể tự định nghĩa các scope cho riêng mình, tuy nhiên các scope cung cấp bởi spring dùng tốt và đáp ứng hầu hết các trường hợp.

***10. @Autowire là gì?***

**- @Autowired** là một annotation của Spring, nó đánh dấu rằng biến này sẽ được init bởi bean container của Spring nếu có phần tử tương ứng. Theo mình, sử dụng **@Autowired** tuân thủ Dependency Injection (DI), và đó là tác dụng của nó. Spring khuyến khích bạn thiết kế chương trình tuân thủ theo DI, giúp ta dễ test, bảo trì và nâng cấp hơn khi không phải chính chúng ta trực tiếp init cho những dependency mà thành phần của chúng ta cần, từ đó chúng không bị ràng buộc với nhau.(Vì Vấn đề đặt ra là khi ta thêm 1 bean vào trong Spring thì có rất nhiều khả năng các bean này nó phụ thuộc lẫn nhau – tức là cái thằng bean này sẽ chứa bean kia hay nó lại phụ thuộc hoặc bị phụ thuộc vào bean khác).

***11. @Component có ý nghĩa gì?***

* ***@Component*** – Sử dụng để định nghĩa một khuôn mẫu chung cho toàn bộ dự án. Các bean được tạo ra trong class được đánh dấu với *@Component* thường sẽ được dùng chung cho toàn bộ dự án hoặc cho một [profile](https://shareprogramming.net/spring-profile-la-gi-ban-co-that-su-hieu-ro/) cụ thể.
* ***@Service*** *–*Dùng để đánh dấu các class thuộc tầng service, nắm giữ code xử lý *.*
* ***@Repository*** – Dùng để đánh dấu các class thuộc tầng giao tiếp với database.

- Chúng ta sử dụng **@Component** trên toàn bộ ứng dụng để đánh dấu một class được tạo ra với mục đích là để định nghĩa các bean được quản lý bởi Spring IoC container.

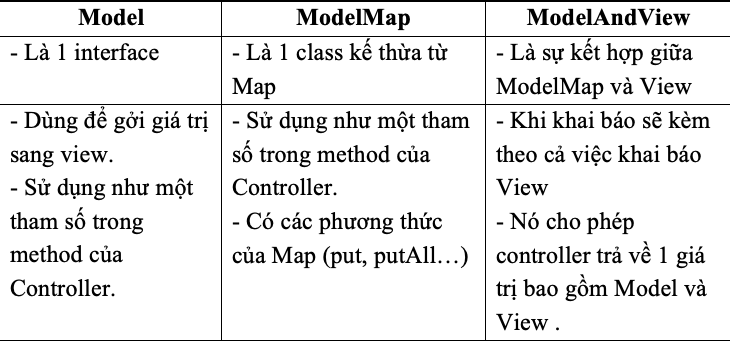
Spring chỉ tìm kiếm và chọn ra những class được đánh dấu với **@Component** annotaion để khởi tạo và đăng ký với **ApplicationContext.**

**-** Về bản chất thì bạn có thể sử dụng 3 Annotation ***@Component,*** ***@Service*** và ***@Repository*** cho nhau mà không ảnh hưởng gì tới code của bạn cả. Nó vẫn sẽ hoạt động. Tuy nhiên từ góc độ thiết kế thì chúng ta cần phân rõ 3 Annotation này cho các Class đảm nhiệm đúng nhiệm vụ của nó.

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

- F**ront Controller** sẽ là nơi đầu tiên tiếp nhận các request từ người dùng và nó cũng là nơi sẽ trả kết quả lại cho người dùng. **Front controller** sẽ điều hướng request tới các controllers phù hợp, các controllers này sẽ xử lý request, trả về model và chọn view sẽ sử dụng (DipatcherServlet).

***13* + *14. Trình bày ý nghĩa của ModelAndView Interface & Trình bày ý nghĩa của ModelMap Interface***



***15. 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

• Có nhiều implementation khác nhau của ViewResolver:

• *AbstractCachingViewResolver*• *XmlViewResolver*• *ResourceBundleViewResolver*

• *UrlBasedViewResolver*• *InternalResourceViewResolver*• *FreeMarkerViewResolver*• *ContentNegotiatingViewResolver*

***16. Phân biệt POST với PUT thường sử dụng để làm gì?***

- Khi chúng ta viết các API thì chúng ta thường sử dụng phương thức POST để tạo dữ liệu còn PUT thì để cập nhật lại dữ liệu. Chúng ta cứ biết vậy và cứ code như thế rồi thấy nó chạy được. Nhưng có lần mình có viết nhầm tạo dữ liệu mình dùng PUT và vẫn thấy tạo được dữ liệu. Nên mình mới tìm hiểu tại sao chúng ta không dùng 1 trong 2 thôi mà phải dùng cả 2 để làm gì?

- Trước hết chúng ta cùng tìm hiểu về sự bảo toàn dữ liệu là gì? Và như thế nào là phương thức bảo toàn dữ liệu? - 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. Bên cạnh đó thì khi tạo dữ liệu mới thì chúng ta đều muốn hệ thống tự tạo id cho dữ liệu đó để không bị trùng lặp id dẫn đến thay đổi dữ liệu. Vì vậy POST để làm việc này chứ không phải là PUT. Vì tạo dữ liệu mới PUT sẽ đi kèm theo với id để hệ thống sẽ tạo id của dữ liệu theo id đó nên sẽ có thể bị gây trùng lặp mà dữ liệu bị thay đổi.

***17. @RequestMapping làm gì?***

- Annotation @RequestMapping được sử dụng để ánh xạ các request tới các action tương ứng của controller.

- ***@RequestMapping*** bao gồm một số thuộc tính để ánh xạ đến: URL, Http Method, các tham số, các header, Media type.

- ***@RequestMapping*** còn có một số biến thể dành cho các HTTP Method cụ thể: ***@GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping.***

***18. 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.

***19. 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

***20. Thymeleaf là gì?***

- Là bộ cộng cụ dùng để xử lý view, dựa trên ngôn ngữ Java (Template engine Java – công cụ mẫu Java)

- Thymeleaf có thể xử lý HTML, XML, JavaScript, CSS và cả text.

- Thymeleaf được xây dựng phù hợp với các tiêu chuẩn của web, đặc biệt là HTML5.

- Những loại template được Thymeleaf hỗ trợ: HTML, XML, TEXT, JAVASCRIPT , CSS, RAW

**- Lợi ích:**

* 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 rendered (kết xuất) 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.

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

**- Vòng lặp: th:each**

Vòng lặp th:each cung cấp các trạng thái sau:

* index: chỉ số của lần lặp hiện tại, bắt đầu từ giá trị 0
* count: chỉ số của lần lặp hiện tại, bắt đầu từ giá trị 1
* size: tổng số các phần tử
* current: phần tử của vòng lặp hiện tại
* even hoặc odd: vòng lặp chẵn hay lẻ
* first: vòng lặp đầu tiên
* last: vòng lặp cuối cùng

**- Biểu thức điều kiện:**

**th:if** trả về true trong các trường hợp:

* Một giá trị boolean là true.
* Một số có giá trị là non-zero (khác 0)
* Một ký tự có giá trị là non-zero (khác 0)
* Một chuỗi có giá trị khác với “false”, “off” hoặc“no”
* Nếu một giá trị không phải là boolean, số, ký tự hoặc chuỗi.

**th:switch…th:case**

* th:case = “\*” ~ default

***22. 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. Chẳng hạn, chúng ta có thể biểu diễn cùng một ngày theo những định dạng khác nhau, chẳng hạn như: *“December 25, 2016,” 12/25/2016,” “2016-12-25”*. Mặc định, Spring sẽ sử dụng định dạng ngày của địa phương hiện tại (current locale). Chẳng hạn, định dạng của Hoa Kỳ sẽ là *MM/dd/yyyy*. Trong trường hợp chúng ta muốn thay đổi định dạng của ngày khi liên kết một trường string với một đối tượng LocalDate thì chúng ta cần viết một converter để chuyển đổi từ string sang ngày.

- Để tạo một converter, chúng ta viết một lớp Java triển khai interface Converter(***pubic interface Converter<S, T>)*** S đại diện cho kiểu dữ liệu nguồn, và T đại diện cho kiểu dữ liệu đích. Chẳng hạn, nếu muốn tạo một converter để chuyển đổi từ kiểu Long(S) sang kiểu LocalDate(T).

- ***Spring*** ***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 ***Formatte****r* 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***.

- Để tạo một ***formatter***, chúng ta viết một lớp Java triển khai interface ***Formatter***(***pubic interface Formatter<T>***). Trong đó, T đại diện cho kiểu dữ liệu đích. Interface này có hai phương thức mà các lớp cần triển khai, đó là *parse()* và *print().* Phương thức parse() chuyển một String sang kiểu dữ liệu đích sử dụng một Locale xác định. Phương thức print() thì thực hiện công việc ngược lại, chuyển đổi từ một đối tượng của kiểu dữ liệu đích sang String.

***23. 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

**-** Là một quá trình ánh xạ (chuyển đổi) dữ liệu từ ngôn ngữ hướng đối tượng sang Database quan hệ và ngược lại. ORM giúp mình ánh xạ các tables,column,kiểu dữ liệu và mối quan hệ (1-1,1-n,n-n) trong database thành các Class và thuộc tính trong Java.

***24. JPA là gì? Spring Data JPA là gì ?***

- ***JPA*** Là đặc tả (giao diện ~ interface) nó sẽ triển khai khái niệm ORM, tức là các table (entity), column, id, CRUD.

- Để nói đến *Spring data JPA* thì ta cần biêt đến khái niệm ***Spring Data,*** nó là một module của Srinng Framework. Mục đích của ***Spring Data*** là giảm thiểu việc thực hiện quá nhiều bước để có thể implement được JPA. Spring Data JPA là một phần của Spirng Data và nó hỗ trợ Hibernate 5 và Hibernate sẽ implement lại JPA hay nói chính xác hơn thì Spring data JPA sẽ giúp cho người dùng thực hiện việc Hibernate implement từ JPA diễn ra nhanh hơn(*Đấy là cách nói dễ hiểu thế*) thực tế ra Spring data JPA sẽ:

+ Cải tiến JPA tiêu chuẩn.

+ Đơn giản hoá tầng truye xuất dữ liệu.

+ Tự tạo Repository.

+ Truy vấn DSL.

+ Ghi Log, phân trang.

+ Có thể tuỳ biến nếu cần thiết.

Nó làm tất cả các điều này thông qua 4 Interface đã học là:

- Repository <T,ID>

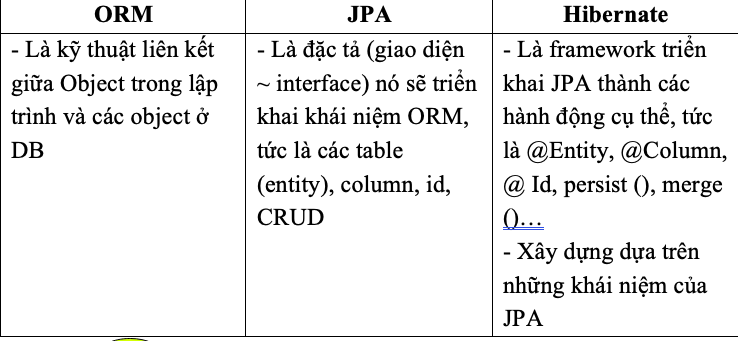
- Crud Repository <T,ID>

- PagingAndSortingRepository <T,ID>

- JPARepository <T,ID> *(cũng có thể phân trang và sort đc do JPARepository <T,ID> kế thừa PagingAndSortingRepository <T,ID>)*

***25. Hibernate là gì ? Hibernate làm gì với database***

Câu này tiện phân biệt luôn như bảng bên dưới.



Hibernate nó làm gì với database ta sẽ thấy thông qua ***“Những lợi ích mà Hibernate mang lại”:***

- **Mã nguồn mở và nhẹ:**Hibernate Framework là mã nguồn mở có giấy phép LGPL và nhẹ.

- **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.

- **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.

- **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.

- **Đơn giản lệnh join phức tạp:** Có thể lấy dữ liệu từ nhiều bảng một cách dễ dàng với Hibernate framework.

- **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.

**\*THIEU CAU 26**

27. ***Các annotation của Hibernate***

* @Entity
* @Table
* @Column
* @Transient
* @Temporal
* @Id
* @GenararedValue

GenerationType.AUTO

GenerationType.IDENTITY

GenerationType.SEQUENCE

GenerationType.TABLE

* @ManyToOne
* @OneToMany
* @ManyToMany
* @OneToOne
* @OrderBy

28. ***Entity là gì ?***

Trong JPA – Hibernate, một entity là một Java class tương ứng với một table trong database, mỗi entity object tương ứng với một dòng trong bảng.

29. ***Persistence Context & Entity Manager là gì ?***

- Persistence context có thể coi là một “môi trường” chưas toàn bộ các đối tượng mà ta tạo ra và lưu vào csdl trong mỗi session. Một sesion hay là 1 phiên, có phạm vi tuỳ vào ứng dụng. Khi ta làm việt 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.

- Entity manager là một interface cung cấp cáp API cho việc tương tác giữa ứng dụng với database :

Môt số chức năng cơ bản của Enity Manager : Persist, Remove

30. ***Câu lênhj truy vấn động là gì ? Câu lệnh truy vấn tĩnh là gì ?***

31. ***Các trạng thái của 1 thể hiện Entity ?***

Trong một session persistence context, các entity có 4 trạng thái :

* Transistent : đối tượng chưa bao giờ bị quản lý bởi sesion và nó không tương ứng với bản ghi nào trong DB, thông thương đây là 1 đối tượng mới được tạo để save vào Db.
* Persistence : đối tượng bị quản lý bởi Session và unique ( duy nhất ), sau khi flush bởi sesion sẽ tồn tại 1 bản ghi tương ứng đối tượng này trong DB.
* Detached : đối tượng này đã từng bị quản lý bởi session nhưng hiện tại thì không.
* Remove : cũng giống như detached nhưng bản ghi tương ứng với đối tượng này trước đó đã bị xoá khỏi DB.

32. ***Có bao nhiêu loại Mapping trong hibernate ?***

Có 3 loại mapping quan trọng : Collection Mappings, Association Mappings (liên kết giữa các lớp entity), Component Mapping .

33. ***SessionFactory là gì ?***

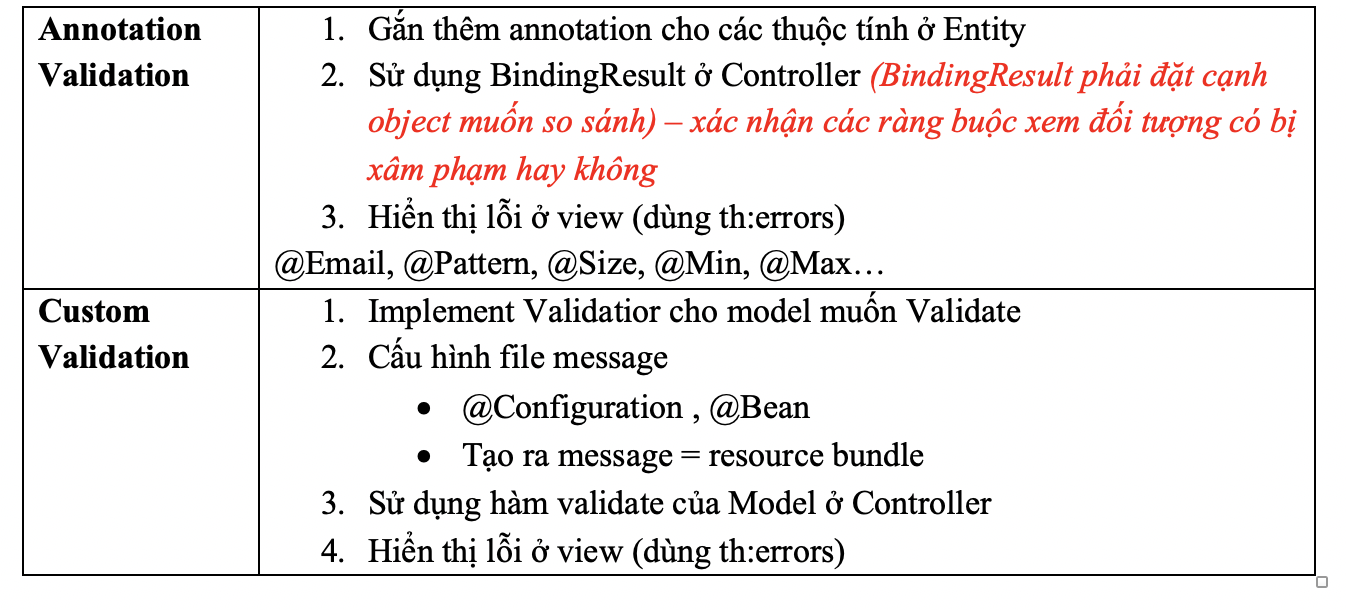
SessionFactory là một đối tượng luồng an toàn và được sử dụng bởi tất cả các luồng của một ứng dụng. Mỗi một DB phải có một session factory

VD : khi ta sử dụng MySQL cho ứng dụng Java của mình thì ta cần có một sesion factory cho MySQL.

34. ***Validation dữ liệu là gì ?***

Bắt lỗi từ input của người dùng nhập vào.

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



36. ***Binding Result là gì ?***

Là một đối tượng của Spring mà giữ kết quả xác thực và ràng buộc, chứa các lỗi có thể xảy ra. BingdingResult phải đến ngay sau khi đối tượng Model được xác thực nếu không Spring sẽ không xác thực và ném các ngoại lệ.

37. ***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.

38. ***Các loại Advise ?***

Trong AOP hỗ trợ cho chúng ta 4 loại Advise

* MethodBeforeAdvice : Chạy trước khi method được chạy
* AfterReturningAdvice : Chạy sau khi method chạy xong
* After Throws Advise : Chạy sau khi method ném ra ngoại lệ
* Around Advice : Tổng hợp 3 cái advice ở trên thành 1.

39. ***JoinPoint là gì ?***

* **Joinpoint**: là một điểm trong chương trình, là những nơi có thể được chèn những cross-cutting concern. Chẳng hạn chúng ta cần ghi log lại sau khi chạy method nào đó thì điểm ngay sau method đó được thực thi gọi là một Jointpoint. Một Jointpoint có thể là một phương thức được gọi, một ngoại lệ được throw ra, hay một field được thay đổi.

40. ***Nêu cách triển khai aop trong dự án ?***

41. ***Xử lý ngoại lệ trong spring web MVC như thế nào ?***

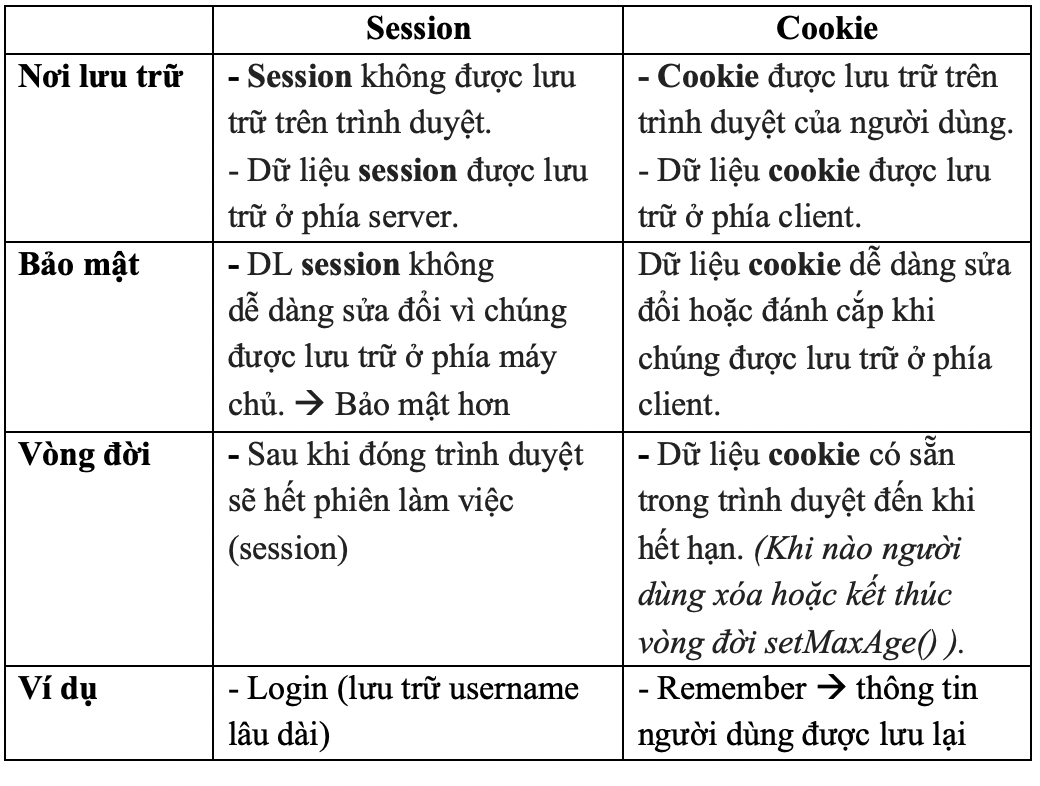
42. ***Cookie là gì ?***

**Cookie** dùng để lưu những thông tin tạm thời.

- **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. Mỗi cookie thường có khoảng thời gian timeout nhất định do lập trình viên xác định trước.

- **Cookie** : kiểu DL chỉ là String

43. ***Phân biệt session và cookie ?***



44. ***Cách triển khai session dùng trong dự án ?***

**- Session:**

- @SessionAttributes thì sử dụng ở đầu Controller nào muốn dùng session.

- @ SessionAttribute thì sử dụng ở tham số ở các method muốn dùng session

45. ***Web service là gì ? lấy ví dụ về web service ?***

**Webservice**là tập hợp các tiêu chuẩn và giao thức được sử dụng để**trao đổi dữ liệu**giữa các ứng dụng hoặc các hệ thống khác nhau**.**

**Ưu diểm :**

**-** Hoạt động trên các ứng dụng, nền tảng, hệ điều hành, ngôn ngữ khác nhau

**-** Khả năng tái sử dụng cao.

-Giảm sự phức tạp của hệ thống, giảm thời gian phát triển hệ thống, hạ giá thành hoạt động, dễ dàng tương tác giữa các hệ thống với nhau.

**Nhược điểm :**

- Khi một web service chết hoặc dừng hoạt động sẽ gây lỗi, thiệt hại lớn trên tất cả các hệ thống, thiết bị đang sử dụng web service đó

-vấn đề an toàn và bảo mật

- Việc có quá nhiều chuẩn cho web service dẫn đến người sử dụng khó nắm bắt.

***\* Ví dụ*:**

**3 ngân hàng sử dụng 3 loại ngôn ngữ lập trình khác nhau và triển khai trên 3 nền tảng OS khác nhau. Webservice nó giúp chúng ta có thể trao đổi dữ liệu từ các ứng dụng, ngôn ngữ, OS khác nhau có thể nói chuyện được với nhau. dựa vào đặc tả webservice thì mình sẽ gọi lên ngân hàng đó và lấy kết quả về cho ứng dụng của mình. Sau đó mình làm gì tiếp với dữ liệu là phụ thuộc vào nghiệp vụ của ứng dụng mình.**

***Câu 46: Phân biệt Web Service và Web truyền thông***

|  |  |
| --- | --- |
| **Website** | **Webservice** |
| - Output sẽ là view (tạo từ Front End: HTML, JS, CSS…) | - Output sẽ là JSON/xml  **(hiện nay thì dùng json)** |
| - Có giao diện người dùng  **(template để hiển thị)** | - Không có giao diện người dùng  **(dùng post main để test)** |
| - Nói đến Website được hiểu sử dụng bởi con người  **(trang web có hình ảnh ,text , … người dùng xem có thể hiểu)** | - Nói đến Web Service được hiểu sử dụng bởi các ứng dụng được tương tác với nhau qua internet  **(trao đổi dữ liệu giữa các ứng dụng hoặc hệ thống khác nhau,**  **Người dùng khó hiểu)** |
| - Website hoạt động đa nền tảng,  **( vì có thể chạy trên nhiều hệ điều hành và trình duyệt khác nhau)** | -Web Service độc lập về nền tảng và tất cả dạng truyền thông đều sử dụng giao thức chuẩn.  **(Web service hoạt động một cách độc lập không phụ thuộc bất kỳ ngôn ngữ nào. Các ứng dụng java, .net hoặc PHP)** |
| - Website được truy cập bởi các thành phần trong giao diện người dùng như button, textbox, form ...  (**ví dụ : các trang web nghe nhạc , trang báo, bán hàng ,wiki…**) | - Webservice được truy cập bởi phương thức HTTP – PUT, GET, POST, DELETE ...  (**ví dụ : google map,các app ,ứng dụng mua sắm ,bank trên đt**) |

|  |
| --- |
| ***47) REST là gì? RESTful là gì?***  **REST**là viết tắt của từ **(Representational State Transfer )**  **1,** State : có nghĩa là trạng thái của dữ liệu từ Object Java mình chuyển sang trạng thái khác để truyền đi  **2,** Transfer : nghĩa là động tác chuyển đổi dữ liệu Object sang kiểu định dạng mới là json hay xml  3, Representational : có nghĩa là hiển thị kiểu json hay xml.  **Restful Webservice**là một dạng webservice viết theo chuẩn REST. REST quy định các quy tắc để bạn làm ra một webservice . Nó chú trọng vào việc lấy resource (tài nguyên) như là data, image , files từ server trả về client thông qua protocol http.  Bất kỳ một ứng dụng nào cũng thực hiện thao tác CRUD (tạo, đọc, sửa, xóa) dữ liệu. Rest đặt ra một quy tắc mà các lập trình viên muốn xác định rõ ý định của mình thông qua các phương thức http.  **Ví dụ sau là một Restful của Spring . Ta định nghĩa các @GetMapping tương ứng với method request là GET. @PostMapping ứng với request là http Post. @DeleteMapping ứng với request có http là DELETE. Và @PutMapping ứng với request Update.** |
| ***48) jQuery là gì? Ajax là gì?***  - **jQuery** là một thư viện kiểu mới của JavaScript ***(Write less, do more - Viết ít hơn, làm nhiều hơn****.)*  Lợi ích :  **1,** làm đơn giản hóa việc truyền tải HTML, xử lý sự kiện, tạo hiệu ứng động và tương tác Ajax.  2**,** là một bộ công cụ tiện ích JavaScript làm đơn giản hóa các tác vụ đa dạng với việc viết ít code hơn.  **3,** một công cụ cung cấp một tập hợp các phương thức AJAX để phát triển ứng dụng web.  **- AJAX** (*Asynchronous JavaScript and XML)*: là **kỹ thuật** xử lý bất đồng bộ của jQuery(**bất đồng bộ nghĩa là cho phép thực thi các đoạn code không theo tuần tự bằng nhưng sự kiện …**)  Mục đích :   1. Tải dữ liệu từ Server mà không cần Refresh trình duyệt 2. Load DL mà không cần phân trang   **- Ví dụ:** bình luận facebook, người A và người B cùng bình luận, không cần load lại trang vẫn hiển thị bình luận 🡪 tự động load theo thời gian  Google dịch : trả nghĩa của từ mà không cần load lại trang. |
| ***49) Nêu ví dụ về cách bạn triển khai Ajax trong dự án của bạn***  $("#btnSearch").click(function () {  $.ajax({  type: "GET",  url: "/search\_blog?keyword=" + $("#keySearch").val(),  dataType: "HTML",  success: function (data) {   $('#listBlog').html(data); |
| ***50) @RequestHeader và @ResponseHeader có ý nghĩa gì*** |
|  |

**@RequestHeader được sử dụng khi ta muốn lấy dữ liệu được truyền bằng Header của một request (yêu cầu từ client)**

**Ví dụ sau ta truyền thêm biến my-number trong phần header của request gửi lên server. @RequestHeader được khai báo trong phương thức doubleNumber có nhiệm vụ lấy giá trị từ header truyền vào biến**

|  |  |
| --- | --- |
| 1  2  3  4  5 | @GetMapping("/double")  public ResponseEntity<String> doubleNumber(@RequestHeader("my-number") int myNumber) {  return new ResponseEntity<String>(String.format("%d \* 2 = %d",  myNumber, (myNumber \* 2)), HttpStatus.OK);  } |

**@ResponseHeader**

**Chúng ta sử dụng @ResponseHeader khi mình muốn trả về thêm dữ liệu cho client ở phần trên cùng của mỗi response**

**Ví dụ sau ta trả thêm các giá trị ở trên phần header cho client thông qua phương thức response.setHeader**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public String addUser(@Valid User user, BindingResult bindingResult,HttpServletRequest request,HttpServletResponse response)  {  if(bindingResult.hasErrors())  {  bindingResult.getFieldError();  return"edit";  }  response.setHeader("Cache-Control","no-cache,no-store,must-revalidate");  response.setHeader("Pragma","no-cache");  response.setDateHeader("Expires", 0);  return "redirect:/welcome/profile/"+user.getName();  } |

***51) I18n và L10n là gì?***

- I18n hay Internationalization 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.

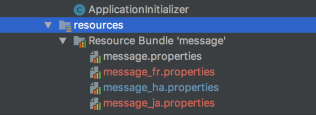
- L10n hay Localization là quá trình điều chỉnh phần mềm đã được quốc tế hóa cho một ngôn ngữ hoặc một khu vực cụ thể bằng cách chỉ định ngôn ngữ hoặc khu vực sau đó dịch văn bản.

* Sự kết hợp giữa i18n và l10n đôi khi được gọi là Globalization và được gọi tắt là g11n

***52) Nêu cách triển khai I18N và I10N trong dự án Spring***

Cần có 1 ResourceBundle tuân theo các quy tắc sau :

* Một tập các file properties
* Tuân theo quy ước đặt tên: \_.properties
* Các message có thể có parameter



Thêm annotation @Configuration cho file WebMVCConfig

Set Basename : classpath:i18n/messages

Thêm các biến vào các file properties

***53) 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

***54) Bảo mật là gì? Cơ chế bảo mật trong Spring như thế nào?***

Bảo mật là quá trình bảo vệ cho website của bạn an toàn tuyệt đối, thiết lập các chế độ bảo mật tầng lớp cần phải có. Đây là 1 điều cực kỳ quan trọng khi thiết kế website.

• Spring Security là một dự án nổi bật trong hệ sinh thái Spring, cung cấp các dịch vụ bảo mật toàn diện cho các ứng dụng doanh nghiệp có nền tảng Java EE.

• Spring Security là một framework tập trung vào việc cung cấp khả năng xác thực (authentication) và phân quyền (authorization) cho ứng dụng Java.

***55) 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.

Xác thực cũng còn có nghĩa khác là công nhận nguồn gốc của đối tượng. Để tiến hành xác thực thì thông thường người ta sẽ sử dụng nhiều nhân tố khác nhau để chứng minh.

* 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.

***56) Liệt kê các cơ chế xác thực***

* HTTP Basic
* Cookies
* Tokens
* Signature
* One-time password
* Certificate
* OAuth2

***57) CSRF 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.

***58) 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.

***59) 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.

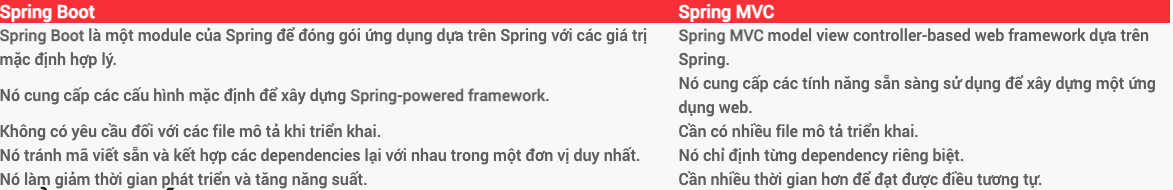
* Cung cấp ra các ứng dụng đã được cấu hình và đóng gói đầy đủ có thể chạy độc lập.
* Giảm thiểu tối đa sự phức tạp của việc cấu hình project cũng như quản lý các thư viện được cung cấp từ bên thứ 3 ngoài Spring.

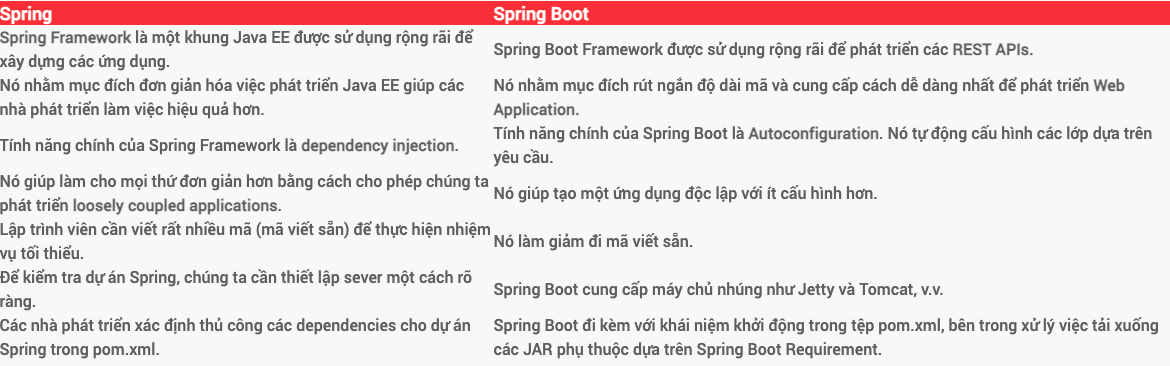
***60. Sự khác nhau giữa SpringBoot và SpringMVC là gì?***

- **Spring Boot** là một module nằm trong Spring Framework, nó cung cấp giao diện và khả năng phát triển các ứng dụng độc lập với rất ít các bước cấu hình rườm rà hoặc gần như bằng không. Spring Boot được đóng gói với rất nhiều thư viện phụ thuộc các module nền tảng của Spring Framework nhưng được giảm thiểu đi các mã nguồn dài dòng, phức tạp nhằm cung cấp sự thuận tiện và phù hợp với từng mục đích khi phát triển ứng dụng.

- Cũng giống như **Spring Boot**, **Spring MVC** cũng là một module nằm trong Spring Framework nhưng mục đích chính của nó là xây dựng các ứng dụng web dựa trên mô hình **MVC** (Model – View – Controller). Spring MVC sử dụng rất nhiều các cầu hình từ Spring do đó cũng chưa rất nhiều các file cấu hình tùy các mục đích khác nhau trong ứng dụng của bạn và nó cung cấp khung phát triển ứng dụng web theo hướng **HTTP**.

*\*Dưới đây là 2 bảng so sánh. Có cung cấp thêm Spring vs Spring Boot để bỗ trợ thêm kiến thức.*



***61. Cấu hình cho Spring Boot Tìm kiếm các Bean ở nhiều package khác nhau bằng cách nào?***

- Về câu này thì mọi người đọc để dễ hiểu hơn nhé. Vào link này để đọc <https://viblo.asia/p/spring-boot-5-component-scan-la-gi-3P0lP4RGlox>

***62. Trình bày Spring Security? @EnableWebSecurity làm gì?***

- ***Spring Security*** là một dự án nổi bật trong hệ sinh thái Spring, cung cấp các dịch vụ bảo mật toàn diện cho các ứng dụng doanh nghiệp có nền tảng Java EE. Spring Security là một framework tập trung vào việc cung cấp khả năng xác thực (authentication) và phân quyền (authorization) cho ứng dụng Java.

- ***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. Còn trong *Spring Secutrity* thì khái niệm này 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.

- ***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. Còn trong *Spring Secutrity* thì khái niệm này là tiến trình quyết định xem một principal *có được phép* thực hiện một hành động trong ứng dụng hay không. Trước khi diễn tiến tới Authorization, principal cần phải được thiết lập bởi Authentication.

-  ***@EnableWebSecurity*** để kích hoạt những dự án Spring Security trong dự án và thường được sử dụng cùng với *@Configuration* để loại bỏ cấu hình bảo mật ứng dụng web mặc định và thêm cấu hình bảo mật của riêng dự án (Hay nói cụ thể là nếu cài đặt *@Configuration, @EnableWebSecurity* ở bất kì nơi nào trong ứng dụng web, nó sẽ tắt cài đặt bảo mật webapp mặc định trong SpringBoot luôn nhưng vẫn bật tính năng bảo mật Actuator Nott).

***63. Trình bày Spring Boot JPA? Cài đặt?***

- **Spring Boot JPA** là một phần trong hệ sinh thái Spring Data, nó tạo ra một layer ở giữa tầng service và database, giúp chúng ta thao tác với database một cách dễ dàng hơn, tự động config và giảm thiểu code thừa thãi.

***\*Cài đặt***

- Để thêm Spring JPA vào project, bạn cần thêm dependency spring-boot-starter-data-jpa. Ngoài ra, để connect tới MySql, chúng ta cần driver tương ứng, vì vậy phải bổ sung thêm cả dependency mysql-connector-java vào pom.xml.

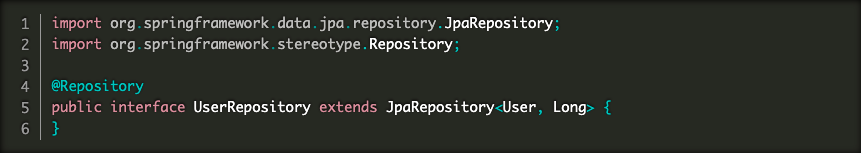
- Để coi ví ụ để dễ hiểu hơn thì mọi người xem tại đây: <https://techmaster.vn/posts/36182/spring-boot-11-huong-dan-spring-boot-jpa-mysql>

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

- Yêu cầu của interface này đó là bạn phải cung cấp 2 thông tin:

+ Entity (Đối tượng tương ứng với Table trong DB).

+ Kiểu dữ liệu của khóa chính (primary key).



***65. Bạn có biết annotation @ConfigurationProperties? Trình bày?***

- Spring boot ra đời và phát triển cung cấp rất nhiều tính năng tiện lợi và một trong số đó là cho phép chúng ta dễ dàng cài đặt và truy xuất các cấu hình trong các tệp cấu hình như  application.properties hay application.yml**.** ***@ConfigurationProperties annotation*** cho phép truy xuất các thuộc tính được đặt trong các tệp cấu hình một cách tự động và mạnh mẽ.

- Mọi người đọc kỹ hơn tại đây:

<https://shareprogramming.net/huong-dan-su-dung-configurationproperties-trong-spring-boot/>