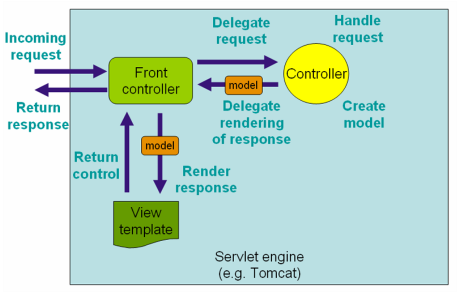
1. Trình bày cách Spring MVC xử lý request



Request từ client gửi đến server được Front controller tiếp nhận (đại diện Front Controller là Dispatcher Servlet). Sau đó Dispatcher Servlet sẽ chuyển đến Controller và Handling Mapping sẽ quyết định method nào của Controller sẽ thực hiện yêu cầu (@Controller và @RequestMapping, @GetMapping, @PostMapping). Sau đó Controller xử lý nghiệp vụ và sẽ trả lại 1 view thông qua ViewResolver (Model/ModelMap/ModelAndView) cho Front Controller (Dispatcher Servlet). Dispatcher Servlet sẽ trả lại client view template nhờ ViewResolver (ánh xạ tên view đến view tương ứng).

1. Trình bày cơ chế Dependency Injection trong Spring

DI là cơ chế tiêm phụ thuộc, tức là tiêm các bean vào trong chương trình viết bằng mô hình Spring MVC, để hạn chế (nới lỏng) sự phụ thuộc giữa các module (các tầng), chỉ phụ thuộc vào abstraction (interface). DI là một Design Pattern triển khai nguyên lý thiết kế phần mềm IOC (Inversion of Control, trong khi IOC cũng là một trong những nguyên lý phát xuất từ nguyên lý cốt lõi trong SOLID đó là DIP – Dependency Inversion Principle – Nguyên lý đảo ngược phụ thuộc). IOC là nguyên lý 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 có trước 🡪 Framework).

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

Có 3 cách để thực hiện cơ chế Dependency Injection (DI). Đó là:

1. Sử dụng Constructor (Interface interface){}
2. Sử dụng phương thức Setter (this.interface = interface);
3. Sử dụng field (thuộc tính): private Interface interface;
4. Framework là gì? Framework khác Library chỗ nào?

Framework là ứng dụng phần mềm có tính trừu tượng cao, nó cung cấp những tính năng chung thông dụng, có sẵn và có thể tùy biến được.

Sự khác nhau cơ bản giữa Framework và Library là cơ chế IOC: Framework nắm quyền điều khiển chương trình/ứng dụng (Code Framework sẽ gọi code của bạn) trong khi Library thì được điều khiển bởi ứng dụng/chương trình (Code của bạn sẽ gọi code Library).

1. Spring Framework là gì?

Là nền tảng mã nguồn mở để phát triển những ứng dụng web Java EE theo mô hình MVC.

Spring gồm có:

1. Front Controller (Đại diện là Dispatcher Servlet) : lắng nghe và chuyển hướng request.
2. Controller: xử lý nghiệp vụ, thao tác ở các tầng Repository (Data) và Service (nghiệp vụ). Trả lại view kèm với data model.
3. ViewResolver: ánh xạ tên view tương ứng với đối tượng view cụ thể.
4. Lợi ích của Spring Framework?

+ Sử dụng được và có thể tùy biến các công cụ/chức năng có sẵn.

+ Tạo sự đồng nhất và đồng bộ code khi phát triển những dự án lớn.

+ Hạn chế sự phụ thuộc lẫn nhau giữa các tầng, từ đó dễ phân chia công việc và thực hiện công việc song song nhau.

+ Dễ bảo trì, dễ mở rộng.

+ Ít quan tâm tới Database nên hạn chế lỗi SQL.

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

Là nguyên lý thiết kế dựa trên nguyên lý thứ 5 trong 5 nguyên tắc thiết kế phần mềm SOLID – Dependency Inversion Principle – Nguyên lý đảo ngược phụ thuộc, tức là để cho Framework thực hiện quyền điều khiển ứng dụng (các đối tượng, các thành phần của hệ thống). Framework sẽ gọi code của lập trình viên khi cần thiết. Trong đó Tiêm phụ thuộc (Dependency Injection) là một Design Pattern triển khai cơ chế IOC này.

1. Bean là gì?

Ám chỉ một đối tượng được khởi tạo và được quản lý bởi Spring (IOC container).

Để đánh dấu một Class là 1 Bean thì thì dùng annotation (@Component, @Service, @Repository, @Controller, @RestController).

1. Trong Spring có bao nhiêu Bean Scope?

Cấu hình bằng @Bean, @Scope(value= constant)

* Singleton: mặc định của Spring, các request dùng chung 1 bean.
* Prototype: Spring container trả về mỗi instance khác nhau cho mỗi request.
* Request: Spring container sẽ tạo một instance cho một HTTP request.
* Session: Spring container sẽ tạo một instance cho một HTTP session.
* Application: Application scope sẽ tạo một bean instance cho lifecycle của một ServletContext.

Application scope thì cũng giống như singleton-scope, tuy nhiên có sự khác biệt quan trọng là scope của application-scope sẽ được shared giữa multipe servlet-based running trên cùng một ServletContext trong khi singleton-scope thì scoped cho một application context.

* Websocket: WebSocket-scope bean được khởi tạo và lưu trữ tại WebSocket session attributes. Instance return là giống nhau khi có request từ WebSocket session.

1. @Autowired là gì?

@Autowired là đánh đấu 1 Bean được tạo và quản lý bởi IOC container (vị trí tiêm Bean).

1. @Component có ý nghĩa gì?

@Component để đánh dấu một class được tạo Bean không thuộc Controller/Service/Repository.

1. Trình bày ý nghĩa của Controller

Controller có ý nghĩa: lắng nghe request, điều hướng và xử lý request.

1. Trình bày ý nghĩa của ModelAndView Interface

ModelAndView interface đại diện cho một view cùng với các dữ liệu sử dụng trong view đó.

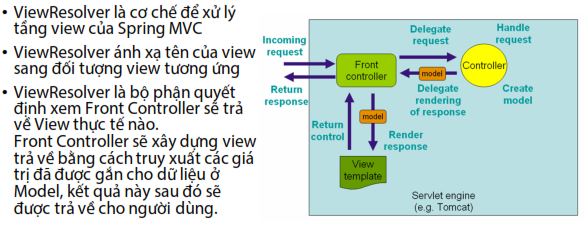
Nó là một container chứa cả view và cả model data ở trong 1 kết quả view trả về.

1. Trình bày ý nghĩa của ModelMap Interface

ModelMap trả về kết quả là data model trong cùng 1 view nhưng ModelMap là interface kết thừa từ Map.

1. Trình bày ý nghĩa của ViewResolver Interface

ViewResolver Interface là trình phân giải view, có nhiệm vụ là ánh xạ tên view (được trả về từ Controller) sang đối tượng View tương ứng với tên view đó.



1. Phân biệt POST với GET thường sử dụng để làm gì?

POST thường dùng để xử lý request tạo mới tài nguyên hoặc cập nhật dữ liệu, vì dữ liệu gửi bằng POST không hiển thị trên URL nên có khả năng bảo mật hơn, dữ liệu gửi đi không bị hạn chế.

GET thường dùng ở request để lấy dữ liệu hiển thị hoặc xóa tài nguyên.

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

POST thường dùng để tạo mới tài nguyên hoặc cập nhật tài nguyên ở trong ứng dụng Web thường. Trong WebService thì POST chỉ dùng để tạo mới tài nguyên.

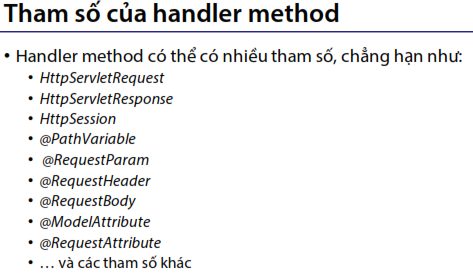
PUT để cập nhật trạng thái của dữ liệu trong WebService thôi.

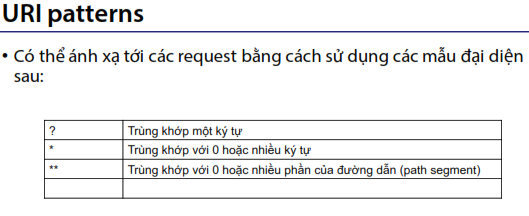
1. @RequestMapping làm gì?

@RequestMapping là ánh xạ các request người dùng từ URL tới các action tương ứng (Handler Method tương ứng) ở trong Controller. @RequestMapping dùng cho cả Class và Method (ở trong tầng Controller).

Các biến thể của @RequestMapping: @PostMapping, @GetMapping, @PutMapping, @DeleteMapping, @PatchMapping.

Một số tham số của Handler Method:





1. 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.

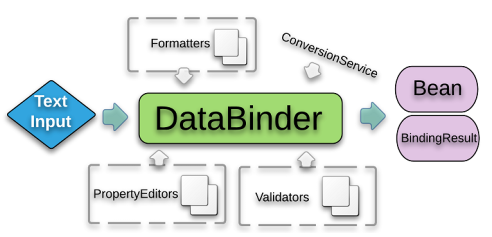
Data Binding giúp cho việc tương tác với dữ liệu trở nên dễ dàng.

Các Form đều được liên kết với một đối tượng dữ liệu được biểu diễn ở phía sau nó.

Khi tương tác với Form, dữ liệu nhập vào Form sẽ được tự động chuyển đổi thành thuộc tính của đối tượng mà nó liên kết.

Data Binding còn hỗ trợ Data Conversion dữ liệu và Validate dữ liệu.

Data Binding trong Spring hoạt động dựa trên Data Binder



1. Thuộc tính consumes trong các Request Mapping là gì ?

Thuộc tính ‘consumes’ trong @RequestMapping quy định Content-Type mà Request có thể gửi lên Server (để Controller tiếp nhận và xử lý).

1. Thymeleaf là gì?

Là một Java Template Engine có chức năng xử lý view cho các ứng dụng web, hoặc các ứng dụng độc lập.

Ưu điểm của Thymeleaf là thân thiện, dễ sử dụng vì nó có thể sử dụng trực tiếp trên các thẻ HTML.

Cú pháp của Thymeleaf là một attribute của thẻ HTML, và bắt đầu bằng [**th:]**

**${…}:** lấy giá trị biến Java trả về.

**@{…}:** lấy địa chỉ URL

**~{…}:**  lấy từng phân đoạn URL.

**\*{…}:** lấy thuộc tính của đối tượng **th:object**

**#{…}:** lấy message

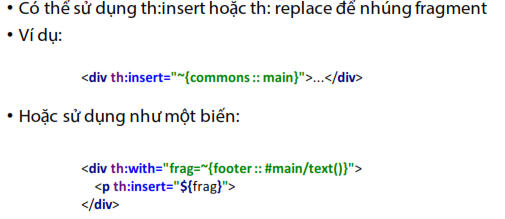
Thymeleaf xử lý được 6 loại template: HTML, XML, TEXT, JAVASCRIPT, CSS, RAW.

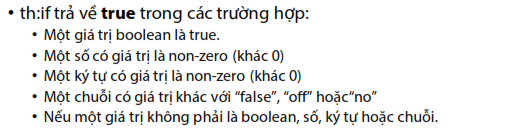
1. Sử dụng lặp trong Thymeleaf như thế nào? Sử dụng điều kiện trong Thymeleaf như thế nào?

+ Vòng lặp Thymeleaf: => th:each= “ele, elStat : ${eList}”

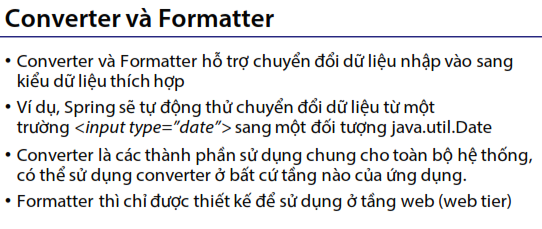
+ Điều kiện Thymeleaf: => th:if và th:unless

+ Switch case Thymeleaf => th:switch và th:case/th:otherwise.





1. Formatter là gì ? Converter là gì ?



1. ORM là gì ?

ORM = Object Relational Mapping: là một kỹ thuật liên kết (ánh xạ) dữ liệu từ đối tượng trong lập trình sang một đối tượng trong cơ sở dữ liệu (tức là quá trình biến đổi 1 record/tuple/row trong DB thành một object trong lập trình OOP.)

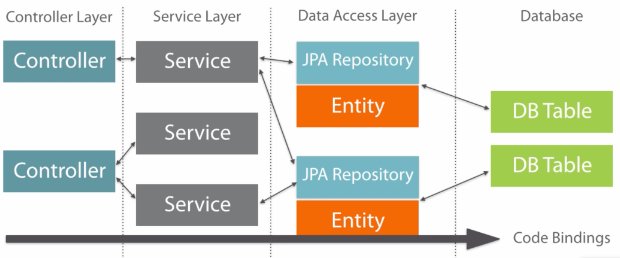
1. JPA là gì? Spring Data JPA là gì ?

JPA – Java Persistence API: là một bộ đặc tả tiêu chuẩn của Java. JPA là một Interface để quản lý các Entity…

JPA mô tả cách quản lý dữ liệu quan hệ trong ứng dụng Java (thông qua Persistence Context và EntityManager).

JPA cung cấp mô hình ánh xạ các bảng trong DB với các Class trong Java.

JPA chỉ là một đặc tả cơ chế ORM, còn việc triển khai các đặc tả đó là Framework Hibernate.



(+) **Ưu điểm của JPA:**

**+** Viết code ít hơn.

**+** Tăng performance của Dev.

**+** Lập trình độc lập với DB.

**+** Không làm việc với SQL nên hạn chế được lỗi phát sinh từ cú pháp SQL.

**+** JPA hỗ trợ triển khai bằng annotation và XML.

1. Hibernate là gì ? Hibernate làm gì với database?

Hibernate là một ORM framework nổi tiếng nhất hiện nay.

Hibernate truy cập DB và sử dụng kỹ thuật ORM để chuyển đổi đối tượng dữ liệu trong DB thành đối tượng trong Java.

**(+) Ưu điểm của Hibernate:**

+ Tăng hiệu suất làm việc (do không làm việc với DB)

+ Thao tác với DL được thực hiện tự động bởi Java.

+ Dễ bảo trì code.

+ Tránh được các lỗi cú pháp SQL (vì không sử dụng SQL trực tiếp).

+ Hỗ trợ giao dịch-transaction.

+ Hỗ trợ cache dữ liệu nên truy xuất dữ liệu được tốt hơn.

(+) **Nhược điểm của Hibernate:**

**+** Chậm hơn JDBC (do phải qua trung gian).

**+** Có thể ảnh hưởng tới hiệu suất của hệ thống (do truy xuất dữ liệu nhiều quá vì tiện quá).

**+** Tốn nhiều thời gian học Framework.

**+** Một số thao tác phức tạp cần tăng performance thì cần dùng SQL thuần.

1. Liệt kê một số annotation của hibernate?

@Entity, @Id, @GeneratedValue, @Unique, @JoinColumn, @JoinTable, @OneToMany, @ManyToOne, @ManyToMany.

1. Entity là gì?

Entity là đối tượng đại diện cho dữ liệu ở trong ứng dụng (có thể là Student, Product, Customer…) tương ứng với 1 record/row/tuple ở trong DB.

Entity phải là POJO (Plain Old Java Object) là các lớp đơn giản, chỉ dùng mô tả dữ liệu, không xử lý logic trong đó.

Entity sẽ được ánh xạ với 1 Table trong DB.

Entity là class Java nên được kết thừa.

Entity hỗ trợ Transaction.

**(+) Điều kiện sử dụng Entity:**

+ Class cần có annotation @Entity của package javax.Persistence.

+ Có constructor không tham số (public).

+ Các biến đối tượng cần khai báo private hoặc protected.

1. PersistenceContext & EntityManager là gì?

Persistence Context là 1 tập thể hiện của Entity được quản lý và tồn tại trong 1 kho dữ liệu.

EntityManager là 1 interface cung cấp các API cho việc quản lý và tương tác với Entity.

Để tạo mới 1 đối tượng của EntityManager thì dùng annotation @PersistenceContext:

@PersistenceContext

EntityManager **em**;

//….

(+) Một số chức năng của EntityManager:

+ persist(entity): lưu 1 thực thể mới.

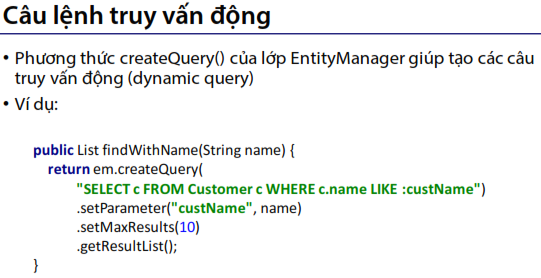
+ merge(entity): cập nhật 1 entity.

+ remove(entity): xóa 1 entity.

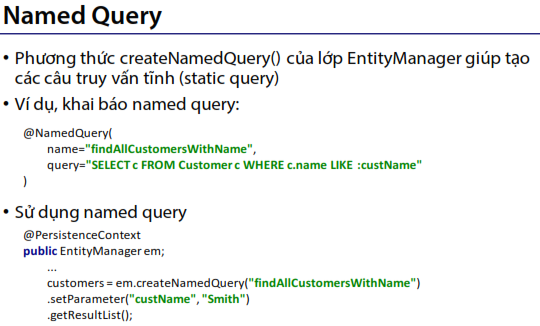
+ find(): tìm kiếm 1 entity trong DB.

1. Câu lệnh truy vấn động là gì? Câu lệnh truy vấn tĩnh là gì?

Câu lệnh truy vấn động là câu lệnh truy vấn cần truyền tham số vào để có thể truy vấn và tái sử dụng được. Câu lệnh truy vấn động được tạo nên bởi phương thức **em.createQuery**()



Còn câu lệnh truy vấn tĩnh thì không cần phải truyền tham số vào, câu lệnh truy vấn tĩnh được tạo bằng cách sử dụng phương thức **createNamedQuery**() của EntityManager:



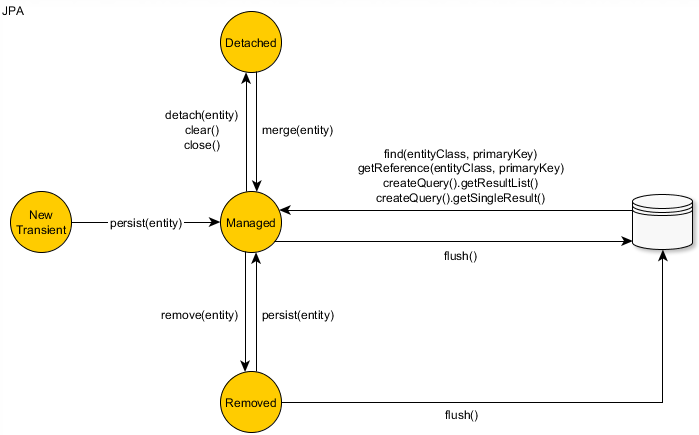
1. Trạng thái của Entity bao gồm những gì?

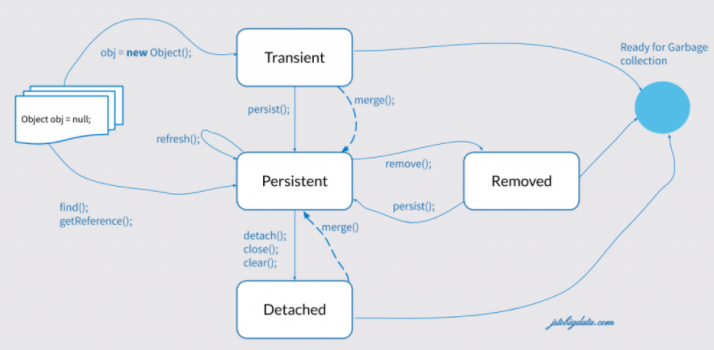
Managed 🡪*sessionFactory.openSession()*

Detatched 🡪Session.evict(entity) or Session.clear()

Transient

Deleted/Removed 🡪Session.delete(entity).





1. Có bao nhiêu loại mapping trong Hibernate?
2. Collection Mappings: Các collection phát sinh từ các mối quan hệ 1-n, n-n
3. Association Mappings: 1-n, 1-1, n-n, n-1
4. Component Mappings: 1 class (không phải là Entity) là thành viên của class khác.
5. SessionFactory là gì?

SessionFactory is an interface. SessionFactory can be created by providing Configuration object, which will contain all DB related property details pulled from either hibernate.cfg.xml file or hibernate.properties file. SessionFactory is a factory for Session objects.

We can create one SessionFactory implementation per database in any application. If your application is referring to multiple databases, then you need to create one SessionFactory per database.

The SessionFactory is a heavyweight object; it is usually created during application start up and kept for later use. The SessionFactory is a thread safe object and used by all the threads of an application.

1. Validation dữ liệu là gì?

Validation dữ liệu là xác minh dữ liệu đầu vào có hợp lệ hay là không.

Chúng ta có thể validate dữ liệu ở cả 3 tầng:

+ UI layer (ở fontend – phía client bằng HTML5/JS/Jquery)

+ Business layer (backend-web server): dùng Java, PHP, C#...

+ Data layer (phía Database): sử dụng constraint trong table hay các câu lệnh SQL. (primary key, not null, unique…)

Một ứng dụng web thông thường sẽ validate ở UI layer và Business layer.

* Spring validation thì validate dữ liệu ở tầng Business layer.

1. Trình bày cách triển khai validate dữ liệu trong Spring

Có 3 cách để validate dữ liệu trong Spring:

1. **Cách 1**: Dùng các annotation có sẵn trong Spring: @Size, @NotNull, @NotEmpty, @NotBlank, @Email, @Pattern(regex = ”^[A-Za-z]\*$”), @Min, @Max, @NumberFormat., @DateTimeFormat, @Past, @Future.
2. **Cách 2**: Dùng custom validate: Class cần validate đối tượng sẽ implements interface **Validator** và ghi đè 2 method supports và validate.

Trong cả 2 cách trên thì trong Controller phải có @Valid/@Validated ở trước @ModelAttribute và tiếp đến là Class BindingResult.

Method validate hoặc annotation trên các trường sẽ gọi Spring check xem có lỗi hay không.

Đối tượng bindingResult sẽ chứa các error xuất hiện (nếu có) và ném trả về view cho người dùng.

1. Sử dụng custom annotation: tạo 1 annotation và 1 class cho Annotation đó. Class này sẽ implements interface **ConstraintValidator<**Tên\_annotation dataType> và ghi đè 2 method**: initialize + isValid(**String value, ConstraintValidatorContext context**)**
2. Binding Result là gì?

Là class chứa các thông tin về việc đối tượng được tạo nên có vi phạm các ràng buộc (constraint) hay không.

Tùy biến thông báo trong trong file \*.properties, hoặc trong Annotation validate luôn. VD : @NotEmpty( message = “Name must not empty”).

1. AOP là gì? AOP gồm những yếu tố nào?

AOP là Aspect Oriented Programming – lập trình hướng khía cạnh, tức là một kỹ thuật lập trình nhằm phân tách chương trình thành các module riêng rẽ, phân biệt và không phụ thuộc lẫn nhau.

VD: chương trình quản lý mượn sách sẽ xử lý nghiệp vụ chính là quản lý sách còn lại và sách đã mượn, sinh viên mượn sách, số lượng sách mượn, mã mượn sách… còn chương trình phụ là chương trình ghi log lại những lần truy cập vào trang mượn sách, hay là những lịch sử trả sách, lịch sử thêm sách mới vào thư viện… hoặc tính năng security, exception handling.

* Những tính năng phụ này được gọi là Aspect, code để thực hiện những Aspect này được gọi là Advice. Các Aspect này tương tự như các class trong Java.
* Chương trình: core concern (primary concern-chương trình chính)

(+) cross-cutting concerns (chỉ mối quan tâm xuyên suốt).

(+) **cross-cutting concern là một khối mã lệnh độc lập có khả năng tái sử dụng. Nó là những xử lý phụ được thực hiện khi core concern của chương trình được gọi lên (**logging, tracking, security, monitoring…)

* Lập trình AOP chính là công việc lập trình các Aspect, tức là viết ra các Aspect và sử dụng chúng.

1. Các loại Advice?

Advice: là những code để thực hiện Aspect, nó là 1 hành động weave advice vào chương trình chính.

Các loại Advice: @After, @Before, @Around, @AfterThrowing, @AfterReturning.

1. JoinPoint là gì?

JoinPoint: điểm gia nhập, là một vị trí nào đó trong luồng thực thi chương trình mà Aspect (cụ thể là Advice) được weave (đan, dệt) vào chương trình chính 🡪JoinPoint = Core Concern x Cross-cutting Concerns.

PointCut: chỉ ra tập hợp các joint point để công cụ AOP (Aspect) đan các advice vào **core-concern**.

1. Nêu cách triển khai AOP mà bạn dùng trong dự án

Cách triển khai AOP dùng trong dự án: Ghi log chương trình mượn sách, trả sách.

Khi người dùng đăng nhập và xem danh mục sách hoặc khi trả sách thì ghi lại log thông tin người mượn, thời gian…

1. Xử lý ngoại lệ trong Spring Web MVC như thế nào?

Mục tiêu của xử lý ngoại lệ (Exception Handling) trong Spring là không gửi thông tin ngoại lệ tới client (người dùng). Mấu chốt của Exception Handling là bắt ngoại lệ đúng lúc và có hướng xử lý (handle) phù hợp.

Các cách xử lý ngoại lệ trong Spring Web MVC:

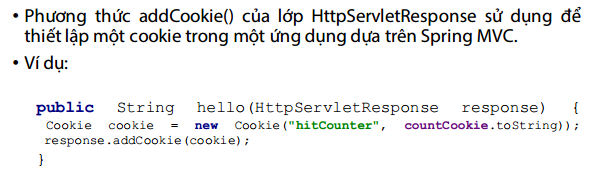
1. Sử dụng try-catch: dùng để xử lý ngoại lệ nghiệp vụ, xử lý tại Controller (hoặc ở Service).
2. Sử dụng annotation @ExceptionHandler: Tạo 1 class ExceptionHandler có annotation là @ControllerAdvice. Class này sẽ có 1 method trả về 1 view với annotation ở đầu method là @ExceptionHandler(BookNotFoundException.class) – ví dụ:

@ControllerAdvice  
**public class** BookNotFoundExceptionHandler {  
  
 *// if exception of this class happens, this method will handle* @ExceptionHandler(BookNotFoundException.**class**)  
 **public** ModelAndView showNotRemainedPage(){  
 System.***err***.println(**"There is no book of this kind remained here..."**);  
 **return new** ModelAndView(**"book/not\_remained"**);  
 }  
}

1. Sử dụng HandlerExceptionResolver: xử lý ngoại lệ hệ thống và từ chối request.
2. Cấu hình error-page.
3. Cookie là gì?

**Cookie:** là các tệp được trang web (website) tạo ra khi người dùng nhập vào. Cookie giúp tăng trải nghiệm trực tuyến người dùng bằng cách lưu thông tin người dùng ở browser. (lấy giá trị của Cookie: cookie.getValue())

@CookieValu sử dụng để truy cập dữ liệu được đặt ở trong bất kỳ http cookie nào.



Ứng dụng: lưu thông tin login của người dùng, ghi nhớ các tùy chọn trong trang web mà người dùng truy cập và cung cấp các nội dung phù hợp…

Trong Spring thì dùng annotation @CookieValue lấy dữ liệu được cài đặt bên trong http cookie.

**Session**: được gọi là phiên làm việc giữa client và server, từ khi client mở browser đến khi đóng browser.

Một session bắt đầu khi client thực hiện request đầu tiên đến server và kết thúc khi client dùng làm việc với server (đóng browser).

1. Phân biệt Session và Cookie
2. Session: lưu trữ thông tin ở Server, trong khi Cookie lưu trữ thông tin ở browser của client.
3. Session kết thúc khi đóng trình duyệt, trong khi Cookie chỉ kết thúc khi hết hạn thời gian sống (setMaxAge()).
4. Session bảo mật hơn vì lưu ở server, trong khi Cookie thì lưu ở browser nên có thể bị đánh cắp thông tin, sửa hay xóa thông tin.
5. Session phụ thuộc vào Cookie (Value “JSESSIONID” là một ID của Session được tạo ra từ Server và gửi về máy client. Khi người dùng đăng nhập browser để gửi request lên server thì server sẽ kiểm tra và so sánh JSESSIONID của Cookie (ở Browser) và JSESSIONID ở server xem thông tin có giống nhau không, nếu có thì server sẽ trả lại thông tin.). Vì thế nên Session phụ thuộc vào Cookie, trong khi Cookie không phụ thuộc vào Session.
6. Nêu cách triển khai Session mà bạn dùng trong dự án

Cách triển khai Session dùng trong dự án: lấy thông tin đăng nhập của User hiển thị lên tất cả các page trong ứng dụng. Hoặc là ứng dụng tạo giỏ hàng thì đối tượng giỏ hàng được lưu ở trong session để có thể chuyển đổi từ trang product list/add sang trang giỏ hàng mà không bị mất dữ liệu.

1. Web Service là gì? Lấy ví dụ về web service

Web Service là dịch vụ web cung cấp nền tảng chung cho nhiều ứng dụng được viết bằng nhiều nền tảng công nghệ khác nhau để có thể giao tiếp được với nhau.

Hoạt động của Web Service được thực hiện dựa trên mô hình client-server, thông qua giao thức HTTPs với **kiểu dữ liệu trả về là JSON/XML.**

**Web Service có 3 thành phần chính (theo dạng XML):**

**+ SOAP:** Simple Object Access Protocol – giao thức trao đổi thông tin.

**+ WSDL:** Web Service Desciption Language – Mô tả thông tin trao đổi.

**+UDDI:**  Universal Description, Discovery, and Integration – Mô tả, xuất bản và tìm kiếm Web Services.

Ví dụ: sử dụng Web Service cho các ứng dụng tạo ExchangeRate (tỉ giá hối đoái, giá vàng), Dự báo thời tiết (Weather Broadcasting) hay là các thông tin chứng khoán, giá nhà đất, các mẫu banner quảng cáo…

1. Phân biệt Web Service và Web truyền thông

Web Service thì không có giao diện, còn Web truyền thông thì có giao diện.

Web Service thì tương tác giữa các ứng dụng với nhau, trong khi web truyền thông là tương tác giữa người dùng và ứng dụng.

Web Service là độc lập với nền tảng (ko phụ thuộc nền tảng) còn web truyền thông thì đa nền tảng (phụ thuộc vào nhiều ngôn ngữ khác nhau, nhiều hệ điều hành khác nhau…)

Web Service thì truy cập thông tin thông qua các method POST, PUT, GET, DELETE..trong khi web truyền thông thì truy cập thông tin bằng form, link, button…

Web Service trả về response là data thuần JSON/XML nên người dùng khó đọc, trong khi web truyền thông trả về các view là văn bản, hình ảnh, video,…

1. SOAP là gì? RESTful là gì?

**SOAP** = Simple Object Access Protocol – giao thức truy cập đối tượng giản đơn, nó là một loại WebService, làm việc với ngôn ngữ XML nên xử lý chậm.

**REST** = *Representational State Transfer –* Chuyển giao trạng thái đại diện, tức là một loại WebService, nhưng thao tác dựa trên đối tượng JSON/XML nên sử dụng tiện hơn (do JSON xử lý nhanh hơn, code dễ gần với Java hơn).

**RESTful** là ứng dụng được triển khai bằng REST nên tuân thủ các quy tắc của REST:

+ Sử dụng giao thức HTTPs, dữ liệu trả về JSON/XML, độc lập nền tảng, sử dụng các phương thức HTTP rõ ràng

+ Phi trạng thái: không có trạng thái như web truyền thông (không lưu thông tin request header…)

+ Hiển thị cấu trúc thư mục như URIs.

+ Linh động, không yêu cầu chặt chẽ như SOAP (XML yêu cầu chặt chẽ).

1. jQuery là gì? Ajax là gì?

Jquery là thư viện được xây dựng từ JavaScript, với phương châm “write less, do more” – Jquery giúp cho Dev viết code ít hơn, làm đc nhiều việc hơn.

AJAX: Asynchronous JavaScript and XML – là một kỹ thuật giúp phát triển các trang web động mà không cần phải tải lại trang (reload page), nó sử dụng cơ chế bất đồng bộ, giúp cập nhật lại nội dung 1 phần của trang mà không cần reload page, làm cho web mượt mà hơn.

* Giúp người dùng tiếp tục sử dụng trang web trong khi client đang request thông tin từ server.

1. Nêu ví dụ về cách bạn triển khai Ajax trong dự án của bạn

Lưu dữ liệu từ trang create mà ko cần phải chuyển trang hay load lại trang.

Tải nội dung thêm khi click vào nút “xem thêm” mà không cần phải tải lại toàn bộ trang.

1. @RequestHeader và @ResponseHeader có ý nghĩa gì?

**The @RequestHeader annotation is used to bind**[**HTTP request header**](https://www.logicbig.com/quick-info/web/http-header.html)**attributes values to controller method parameters.**

*@GetMapping("/greeting")*

*public ResponseEntity<String> greeting(@RequestHeader("accept-language") String language) {*

*// code that uses the language variable*

*return new ResponseEntity<String>(greeting, HttpStatus.OK);*

*}*

1. I18n và L10n là gì?

Internationalization, Localization

Internationlization: quốc tế hóa, là chức năng setting trang web cho việc sử dụng đa ngôn ngữ và tăng tính thương mại, mở rộng nhiều thị trường khác nhau.

Localization: phán định ngôn ngữ địa phương để hiển thị phù hợp với từng vùng hay từng tùy chọn người dùng.

1. Nêu cách triển khai I18N và L10N trong dự án Spring

(a). Tạo LocaleChangeInterceptor: Interceptor để gắn biến “language” vào đuôi request để LocaleResolver căn cứ vào biến đó để phán định ngôn ngữ phù hợp.

(b) Setting LocalResolver: Tạo LocalResolver để phán định ngôn ngữ từ request gửi đi theo biến “language” mà Interceptor đã gắn.

(c) Tạo các file message bundle tương ứng với từng ngôn ngữ => với các hậu tố do người dùng quy định

1. Interceptor là gì?

Interceptor là bộ lọc đầu vào của request (tương tự như Servlet Filter), định nghĩa các hậu tố cho các request.

1. Bảo mật là gì? Cơ chế bảo mật trong Spring như thế nào?

Bảo mật là cơ chế đảm bảo tính bí mật và an toàn cũng như sự toàn vẹn của thông tin và dữ liệu.

Trong Spring thì bảo mật qua 2 cơ chế: xác thực (authentication) và phân quyền (authorization)

1. Authentication là gì ? Authorization là gì ?

+ **Authentication**: Là hành động nhằm thiết lập hoặc chứng thực thông điệp, đối tượng nào đó là đáng tin cậy.

+ **Authorization**: Là quá trình xác định đối tượng sau khi xác thực có quyền truy cập vào tài nguyên nào đó của hệ thống, được phép thực hiện cụ thể hành động nào (CRUD) với tài nguyên đó.

1. Liệt kê các cơ chế xác thực:

* Basic HTTP
* Cookie
* Tokens
* One-Time Password
* Signatures
* Certificate
* Oauth2

1. CSRF là gì?

Cross-Site Request Forgery: là một kỹ thuật tạo các request giả mạo xuyên qua các trang.

1. CORS là gì?

**CORS** (*Cross-origin resource sharing*) allows a webpage to request additional resources into browser from other domains e.g. fonts, CSS or static images from CDN. CORS helps in serving web content from multiple domains into browsers who usually have the [same-origin](https://en.wikipedia.org/wiki/Same-origin_policy) security policy.

1. Spring Boot là gì?

**Spring Boot** là một dự án phát triển bởi **JAV**(ngôn ngữ java) trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển business cho ứng dụng.

1. Sự khác nhau giữa SpringBoot và SpringMVC là gì?

Spring Boot tự động cấu hình và tải các dependency cần thiết cho việc build ứng dụng.

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

@Component, @Service, @Repository, @Controller

Sử dụng @ComponentScan, @SpringBootApplication.

1. Trình bày Spring Security? @EnableWebSecurity làm gì?

Spring Security là 1 dự án cung cấp 2 cơ chế bảo mật cơ bản:

+ Authentication: là tiến trình thiết lập một Principal (1 người, thiết bị, hệ thống) có thể thao tác một hành động nào đó lên ứng dụng.

+ Authorization: là tiến trình cấp phép Principal có được phép thực hiện một hành động (truy cập, CRUD) lên tài nguyên của ứng dụng hay không.

Spring Security gồm có 5 thành phần chính:

1. SecurityContext: là interface lưu trữ tất cả chi tiết liên quan đến bảo mật của ứng dụng.
2. SecurityContextHolder: lưu trữ SecurityContext hiện tại của ứng dụng, bao gồm chi tiết cảu Principal đang tương tác với ứng dụng.
3. Đối tượng Authentication: biểu diễn thông tin của Principal.
4. UserDetails: là 1 interface, đại diện cho 1 Principal.
5. GrantedAuthority: là quyền được cấp cho Principal, tiền tố mặc định của quyền là “ROLE\_”. Phương thức getAuthorities() là lấy danh sách các quyền được cấp cho Principal.

The **@EnableWebSecurity** is a marker annotation. It allows Spring to find (it's a **@Configuration** and, therefore, **@Component**) and automatically apply the class to the global WebSecurity.

The WebSecurityConfig class is annotated with @EnableWebSecurity to enable Spring Security’s web security support and provide the Spring MVC integration. It also extends WebSecurityConfigurerAdapter and overrides a couple of its methods to set some specifics of the web security configuration.

1. Trình bày Spring Boot JPA? Cài đặt?
2. Để sử dụng interface JpaRepository cần cung cấp những thông tin nào?

@Entity, @Id, @GeneratedValue

Validation-api

Hibernate-Validator

1. Bạn có biết annotation @ConfigurationProperties? Trình bày?