MODULE 4

OVERVIEW SPRING MVC

Framework
- Là ứng dụng phần mềm có tính trừu tượng , cung cấp các tính năng chung, thông dụng có thể tùy biến để tạo nên những ứng dụng cụ thể khác nhau.
- Mỗi framework cung cấp các phương pháp riêng biệt để xây dựng và triển khai ứng dụng.
- Mỗi framework bao gồm 1 môi trường tổng thể, tái sử dụng được, nhằm cung cấp các chức năng và công cụ để hỗ trợ quá trình phát triển ứng dụng.

Điểm khác biệt so với Library:
+ Library: lập trình viên sẽ nắm quyền điều khiển (Code của mình sẽ gọi Code thư viện).
+ Framework: Framework sẽ nắm quyền ứng dụng (Code của Framework sẽ gọi Code của mình) .
+ Khi code với framework thì cần tuân theo những nguyên tắc có sẵn.
=> Điểm khác biệt chính giữa Framework và Library là cơ chế IoC.

IOC
Inversion of Controller – "nguyên lý đảo ngược quyền điều khiển", là nguyên lý trong phát triển phần mềm, trong đó việc điều khiển các đối tượng hoặc thành phần của hệ thống được thực hiện bởi framework hoặc các container.

DI
Dependency Injection
– Là design pattern cho phép xóa bỏ sự phụ thuộc.
- Là một cơ chế để triển khai IoC.
- Thao tác tiêm (inject) vào các đối tượng khác được thực hiện bởi container hoặc framework.

Có 3 cách triển khai:
+ Constructor - Dùng inject (tiêm) các module bắt buộc. Các module được tiêm nằm trong constructor, và được gán lần lượt vào các field.
+ Setter method - Dùng inject các module tùy chọn. Mỗi module sẽ được inject thông qua setter, nằm ở tham số và cũng gán cho field nào đó.
+ Interface (field) - Các client phải triển khai một Interface mà có một setter method dành cho việc nhận dependency.

Spring Framework
Là 1 framework cung cấp mô hình đầy đủ để phát triển các hệ thống java lớn, giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử và tái sử dụng.

Luồng thực thi
-> Request từ client sẽ gửi tới Front Controller (đây là điểm duy nhất nhận request từ client và trả respone lại cho client)
-> Tại Front Controller thì các request sẽ được chuyển tới các controller tương ứng thông qua cơ chế mapping mà chúng ta khai báo trong handle mapping
-> Sau khi controller nhận request thì sẽ xử lí các nghiệp vụ logic và sau đó sẽ trả respone về cho front controller
-> Front Controller nhận respone sẽ truyền tới view để tìm view tương ứng và sử dụng view cơ chế ViewResolver để truyền model vào view đó .
-> View sẽ trả về cho front controller và front controller sẽ gửi trả về client để hiển thị cho người dùng.

SPRING CONTROLLER

Cách tạo Controller
Dùng @Controller hoặc @RestController.
Trước phiên bản Spring 2.5 thì implements interface Controller.
- @Controller – dùng để đánh dấu 1 class là 1 Spring MVC controller (dùng cho web application).
- @RestController – dùng đánh dấu 1 class là resfull controller (dùng cho ứng dụng web service).

@Request Mapping
@RequestMapping dùng để ánh xạ các request tới các action tương ứng của controller.

Thuộc tính:
+ value (URL pattern).
+ method: GET, POST, PUT, PATCH, DELETE,...
+ consumes: định dạng kiểu dữ liệu khi client request tới server (tương ứng với cài đặt "content-type" của request header).
+ produces: định nghĩa kiểu dữ liệu trả về cho client (tương ứng với cài đặt "accept" của request header).

Các biến thể :
@GetMapping : lấy tài nguyên.
@PostMapping : tạo tài nguyên.
@DeleteMapping : xóa tài nguyên.
@PutMapping : cập nhật toàn bộ tài nguyên.
@PatchMapping : cập nhật 1 phần tài nguyên.

@RequestParam - @PathVariable
@RequestParam – lấy giá trị theo định dạng key-value (param động – có hay không có key cũng được) -> thường dùng trong form.
Thuộc tính:
+ value: tên của parameter (key) gửi lên server.
+ required: quy định param có bắt buộc hay không (mặc định là true).
+ default: giá trị mặc định của param.

@PathVariable – lấy giá trị tham số trên URL (chỉ lấy value), sử dụng được regex.

@Component - @Autowired - Bean
@Component – là một annotation của class. Nó dùng để đánh dấu class Java là một bean. Khi Spring start thì nó quét qua các annotation có đánh dấu là @Component thì nó sẽ tạo bean cho class đó.

@Autowired – tự động nhúng các bean được Spring Container sinh ra vào Class có khai báo @Autowired.

Bean – là những module chính của chương trình, là 1 đối tượng được quản lý bởi Spring IoC container (do framework tự khởi tạo, tự quản lý vòng đời của đối tượng).

Các cách khai báo:
+ Dùng annotation: @Controller, @Service, @Repository, @Component.
+ Dùng kết hợp @Configuration + @Bean.
+ Khai báo Bean bằng XML.

Có 5 Bean Scope:
+ Singleton – chỉ duy nhất một thể hiện của bean sẽ được tạo cho mỗi container. Đây là scope mặc định cho spring bean.
+ Prototype – khác với Singleton, 1 thể hiện của bean sẽ được tạo cho mỗi một yêu cầu (request).
+ Request – giống với prototype scope, tuy nhiên nó dùng cho ứng dụng web, 1 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.

DATA BINDING - FORM

Data binding
Data binding – là cơ chế liên kết dữ liệu đầu vào / đầu ra với đối tượng model.
- Giúp làm việc với form đơn giản hơn.
- Các giá trị ở trong thẻ form sẽ tự động chuyển đổi thành giá trị của thuộc tính trong model.
- Hỗ trợ chuyển đổi dữ liệu và validate.

Hoạt động dựa trên data binder.
+ Mỗi đối tượng như là 1 form, mỗi thuộc tính tương ứng với 1 ô input.
+ Text input sẽ được mapping với data binder thông qua propertyEditors (thông qua getter/setter).
+ ConversionService giúp convert (chuyển đổi) dữ liệu 1 cách tự động.
+ Formatters giúp can thiệp sâu về cấu hình (ví dụ như validate dữ liệu ngày, giờ).
+ Validators giúp validate trực tiếp thuộc tính của đối tượng.
+ Đầu ra sẽ là 1 Bean => gán giá trị vào đối tượng (bắt đối tượng thông qua ModelAttribute).

Form
Thư viện Spring-form-tld hỗ trợ tạo form trong spring, bổ sung thêm các thuộc tính ModelAttribute của thẻ form.
+ Spring-form hỗ trợ các thẻ input, checkbox, radio,...
+ Một số thuộc tính quan trọng:
- path – tên thuộc tính của đối tượng liên kết với form.
- items – tập kết quả muốn hiển thị cho người dùng (thường dùng với select, checkboxs, radio buttons,...).

THYMELEAF

View
Là 1 trong 3 phần của Spring MVC , có nhiệm vụ hiển thị giao diện và tương tác với người dùng
Trong Spring MVC View nằm ở phần Back-End
ViewResolver là cơ chế xử lí tầng view của Spring MVC , ánh xạ tên của view sang đối tượng view tương ứng.
ViewResolver là bộ phận quyết định Front Controller sẽ trả về view nào cho client.

Thymeleaf
Thymeleaf – là 1 bộ xử lý view, được sử dụng cho các ứng dụng web và các ứng dụng độc lập.
+ Được sử dụng cho phù hợp với tiêu chuẩn web, hỗ trợ HTML 5.
+ Thymeleaf cho phép hỗ trợ các template: HTML, XML, JS, CSS, TEXT,RAW.

Các biểu thức:
+ Biểu thức với biến ${...}.
+ Biểu thức với thuộc tính *{...}.
+ Biểu thức với message #{...}.
+ Biểu thức với URL @{...}.
+ Biểu thức với phân đoạn ~{...}.

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: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 / odd: vòng lặp chẵn / lẻ.
• first / last: vòng lặp đầu tiên / cuối cùng.

@ModelAttribute
@ModelAttribute – nhờ là một cầu nối giữa Controller và View. Từ Controller chúng ta truyền các dữ liệu qua cho View thông qua ModelAttribute. Từ View chúng ta sẽ sử dụng Thymeleaf để đọc các dữ liệu từ model và hiển thị ra cho người dùng.

VALIDATION

Xác minh tính hợp lệ của dữ liệu đầu vào.

Các tầng thực hiện Validation
UI layer
- Phía client.
- Dùng ngôn ngữ front-end (JS, HTML,...).
- Validate trực tiếp, tốc độ phản hồi nhanh, mượt, cân bằng tải (giảm tải cho server).
- Bảo mật không cao, nhiều dữ liệu không xử lý được, có thể F12 để chỉnh sửa dữ liệu.

Business layer
- Phía server.
- Dùng ngôn ngữ server (Java, C#, PHP,...).
- Ràng buộc chặt chẽ, độ chính xác cao, xử lý đa dạng.
- Nhiều xử lý khi có nhiều truy cập => server quá tải.

Data layer
- Phía DB server.
- Dùng SQL để tạo ràng buộc (not null, check, primary,...).
- Cú pháp đơn giản, ngắn gọn, dễ sử dụng.
- Khó bắt hợp lệ, tốc độ chậm (phải xét đến sự liên kết giữa các table – CSDL quan hệ).

Các cách validate :
- Sử dụng các annotation có sẵn
- Sử dụng custom validation
- Sử dụng custom annotation

SPRING BOOT - SPRING DATA REPOSITORY

Spring Boot
Spring boot – là 1 trong số các module của Spring Framework, được sử dụng để tạo ra các ứng dụng độc lập, đơn giản trong việc config.
Ưu điểm:
+ Tạo ứng dụng độc lập, chạy ngay bằng cách sử dụng lệnh Java.
+ Nhúng sẵn server Tomcat.
+ Cung cấp sẵn các thư viện "starter".
+ Tự động cấu hình Spring.
+ Không yêu cầu cấu hình XML.

Spring Data Repository

Spring Data JPA
- Là 1 phần của dự án Spring Data.
- Có thể sử dụng với các ORM framework như Hibernate, Open JPA,...
- Làm việc dễ dàng với tầng Repository.
=> Giúp cải tiến JPA tiêu chuẩn, đơn giản hóa tầng truy xuất dữ liệu, tự tạo repository, tạo các truy vấn JPA thông quá tên phương thức.

Interface trong Spring Data JPA:
+ Repository : quản lý entity, kiểu dữ liệu khóa chính entity.
+ CrudRepository : Repository + các phương thức CRUD.
+ PagingAndSortingRepository : CrudRepository + phân trang và sắp xếp.
+ JpaRepository : tích hợp và cải tiến thêm chức năng của các interface trên.

Pageable - Slice
Đều được sử dụng để hỗ trợ phân trang

Pageable:
- Biết được tổng số trang
- Không thích hợp khi sử dụng cho dữ liệu lớn => Vì cần phải đếm từng record để tính ra được tổng số trang làm chậm ứng dụng.

Slice:
- Không biết được tổng số trang
- Thích hợp sử dụng cho dữ liệu lớn khi phân trang vì không cần phải đếm số record để tính ra tổng số trang.

Các thuộc tính / phương thức thông dụng:
+ number : trang hiện tại (bắt đầu từ 0).
+ totalPage : tổng số trang.
+ hasNext( ) : kiểm tra có trang tiếp theo hay không.
+ hasPrevious( ) : kiểm tra có trang trước đó hay không.

Converter - Formatter
Hỗ trợ chuyển đổi dữ liệu nhập vào sang kiểu dữ liệu thích hợp.

Converter
- Là thành phần sử dụng chung cho toàn hệ thống.
- Làm việc với bất kì kiểu dữ liệu nào.

Formatter
- Sử dụng ở tầng web (databinder).
- Kiểu dữ liệu nguồn là String.

ORM - SPRING DATA JPA

ORM
Object Relational Mapping
Là kĩ thuật liên kết các đối tượng trong lập trình với các đối tượng trong cơ sở dữ liệu
ORM cho phép truy xuất dễ dàng đến DB thông qua các đối tượng lập trình.
ORM giúp lập trình viên tập trung vào nghiệp vụ.

ORM có rất nhiều triển khai (Hibernate, Eclipselink, Ibatis,...), trong đó Hibernate được xem là ORM Framework nổi tiếng nhất và sử dụng rộng rãi trong Java.

Hibernate
Hibernate - là một ORM Framework mã nguồn mở giúp LTV viết ứng dụng Java có thể mapping các Object với hệ CSDL, và hỗ trợ thực hiện các khái niệm OOP với CSDL.
Hibernate có thể hỗ trợ tạo bảng và các dữ liệu thông qua ngôn ngữ Java.

Một số annotation của Hibernate: @Entity, @Table, @JoinTable, @Id, @GenerateValue, @Column, @JoinColumn, @OneToOne, @OneToMany, @ManyToOne, @ManyToMany,...

Các loại mapping trong Hibernate:
+ Association Mappings (hay Entity Mappings) : là ánh xạ các liên kết giữa các lớp entity và các mối quan hệ giữa các bảng chính, gồm 4 kiểu liên kết:
- Many-to-One.
- One-to-Many.
- One-to-One.
- Many-to-Many.
+ Collections Mappings : nếu một entity hoặc lớp có một tập các giá trị trong một trường Collection cụ thể, thì chúng ta có thể ánh xạ các giá trị đó bằng bất kỳ một trong các interface collections có sẵn trong java.
1 số kiểu collection: Set, SortedSet, List, Collection, Map, SortedMap.
+ Component Mappings : mapping cho một lớp có 1 tham chiếu đến 1 lớp khác như là một biến thành viên. Sử dụng phần tử <component> để thiết lập sự tồn tại của các thuộc tính khác của lớp của biến tham chiếu bên trong lớp sở hữu nó.

JPA
Java Persistence API – là 1 bộ đặc tả tiêu chuẩn của Java, trong đó mô tả các thao tác quản lý dữ liệu trong Java cung cấp các mô hình cho phép ánh xạ các bảng trong CSDL với các class.
=> JPA chỉ chứa interface hướng dẫn triển khai ORM, việc implement sẽ do ORM framework thực hiện.

Entity
Là đối tượng đại diện cho dữ liệu trong ứng dụng.

+ Entity thường là POJO (Plain Old Java Object), chỉ mô tả dữ liệu, không có xử lý logic nghiệp vụ.
+ Mỗi Entity được mapping với 1 table trong CSDL (mapping 1-1).
+ Mỗi Entity cần phải tuân thủ:
- Được gắn annotation @Entity, @Id.
- Có Constructor là public không tham số.
- Không được khai báo final.
- Thuộc tính có access modifier là private, protected, hoặc ở mức package private (tức là không phải public).

Persistence Context – là một tập các thể hiện của Entity, được dùng quản lý Entity trong 1 kho dữ liệu.

EntityManager – là 1 interface cung cấp các phương thức để thao tác với entity:
+ persit( ) – thêm 1 entity.
+ merge( ) – chỉnh sửa, cập nhật.
+ remove( ) – xóa.
+ find( ) – tìm kiếm.

Trạng thái của các Entity :
+ 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.
+ 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.
+ 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.
+ Removed (Đã bị xóa): tương tự như detached nhưng bản ghi tương ứng với đối tượng này trước đó đã bị xóa khỏi database.