@Component: Đánh dấu trên các class để Spring biết đấy là 1 Bean

Khi chạy Spring Boot sẽ tìm xem Class nào có @Component đưa vào ApplicationContext để quản lý

@Autowired: Khi 1 Bean được đánh dấu @Autowired Spring sẽ tìm kiếm bean tương ứng và inject nó vào thuộc tính hoặc phương thức được đánh dấu @Autowired

@Scope: Sử dụng để chỉ định phạm vi của Bean. Mặc định các bean sử dụng phạm vi singleton, tức là các Bean được quản lý trong ApplicationContext chỉ được tạo ra duy nhất 1 lần, khi có yêu cầu @Autowired thì sẽ được inject vào. Nếu gọi nhiều lần thì vẫn gọi đến đối tượng đó. Nếu muốn mỗi lần dùng sẽ là đối tượng mới thì ta chuyển thành @Scrope(“prototype”): mỗi lần yêu cầu thì Spring tạo ra 1 đối tượng mới.Ngoài ra còn 1 số giá trị như: request,sesion, globalSession

@Primary: Khi có 2 hay nhiều bean cùng loại trong context,khi gọi ra Spring không biết chọn cái nào thì @Primary dùng để đánh dấu 1 trường hợp luôn được ưu tiên trong context đó, khi gọi ra thì sẽ gọi TH đó

@Qualifier: Xác định tên cho Bean khi đặt tên @Component(“name”), khi gọi đến @Qualifier(“name”)

@Configuration: đánh dấu trên 1 class cho phép Spring biết đây là nơi định nghĩa ra các Bean

@Bean được đánh dấu trên các method cho phép Spring biết đây là Bean và thực hiện đưa Bean vào context

@Bean nằm trong các class có đánh dấu @Configuration

@Congiguretion và @Component về bản chất tương tự nhau. Nhưng nếu 1 Bean có quá nhiều logic để khởi tạo và cấu hình thì ta sẽ sử dụng @Còniguration Và @Bean để tự tạo ra Bean

@Controller dùng để đánh dấu 1 class là Spring MVC Controller. Là nơi chấp nhận yếu cầu và trả về phản hồi cho người dùng, có thể hiểu là trung gian giữa Server và bên ngoài

@RequestMapping: ánh xạ các HTTP request tới các phương thức xử lý của MVC và REST controller.

Cú pháp **@RequestMapping (value, method, params)**

* value: chỉ ra action được ánh xạ với phương thức xử lý
* method: chỉ ra phương thức truyền dữ liệu của trình duyệt web (POST, GET, PUT, DELETE.v.v.)
* params: chỉ ra tham số bắt buộc phải có để thực hiện action.

@GetMapping cho method GET, dùng để lấy tài nguyên, thông tin được hiển thị lên URL

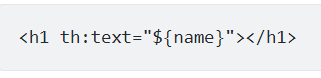
@GetMapping() là rút gọn của RequestMapping(method=”RequestMethod.GET”)

**Thymeleaf**

**Thymeleaf** là một Java Template Engine. Có nhiệm vụ xử lý và generate ra các file HTML, XML, v.v..

Các file HMTL do Thymeleaf tạo ra là nhờ kết hợp **dữ liệu** và **template + quy tắc** để sinh ra một file HTML chứa đầy đủ thông tin.

Cú pháp: Thymeleaf sẽ là 1 attributes của thẻ HTML bắt đầu bằng th:

Ví dụ: 

Model & View trong Spring Boot:

Model là đối tượng lưu trữ thông tin được sử dụng bởi Template Engine để generate ra webpage. Có thể hiểu nó là context của Thymeleaf

Model lưu trữ thông tin dưới dạng key-value

Để lấy thông tin trong Model ta sử dung Thymeleaf Standard Expression

1. ${}: giá trị 1 biến

VD: Trên cotroller đưa vào 1 số giá trị



Để lấy giá trị biến today : <p th:text:${today}”></p>

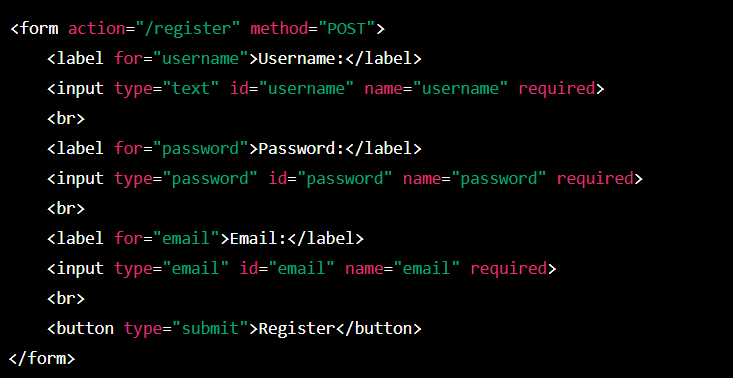
1. \*{} Chức năng giống với ${}, điểm khác biệt là lấy giá trị cho trước bởi th:object,

vd: sử dụng để lấy ra thuộc tính của đối tượng

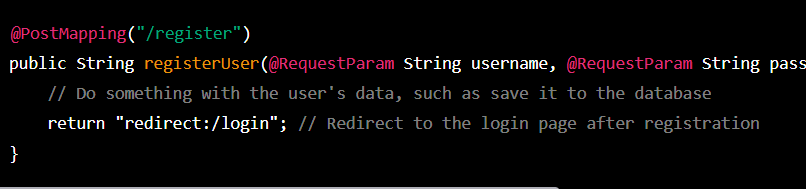
1. @{...} xử lý và trả ra giá trị URL theo context của máy chủ cho chúng ta.

Gửi dữ liệu từ View sang Controller bằng form trong Thymeleaf:

Ta nhập dữ liệu từ form trong html và gửi dữ liệu từ trang web đó đến 1 phương thức POST hoặc GET



Trong Cotroller tạo phương thức:



Sau khi nhâp dữ liệu, bấm nút Submit thì phương thức trên xử lý yêu cầu POST ,

@RequestParam để nhận dữ liệu được nhập và gán với các biến tương ứng