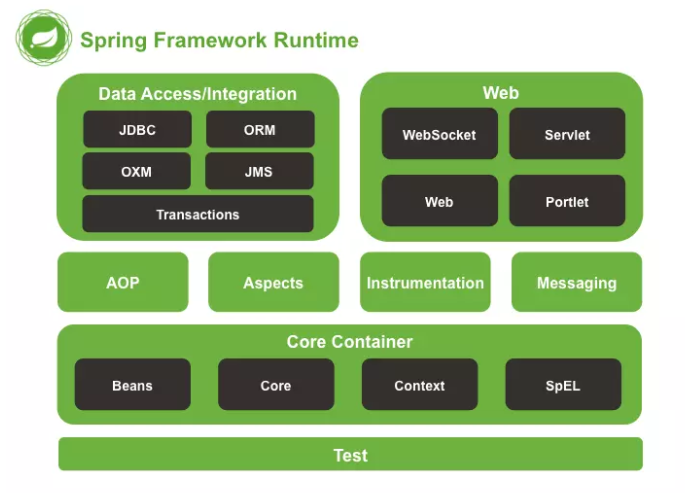
**SPRING VÀ SPRING BOOT**

1. **Tổng quan về Spring**

* Spring là một framework mã nguồn mở (open source), được phát triển và có cộng đồng người dùng rất lớn trong giới lập trình.
* Spring framework được phát triển dựa trên 2 nguyên tắc thiết kết chính là dependence injection và AOP (lập trình hướng khía cạnh)
* Phiên bản đầu tiên được tạo ra bởi Rod Johnson và phiên bản framework đầu tiên thì được phát hành vào 6/2003. Từ đó đến này đã có rất nhiều phiên bản ra đời:
  + Version 1.0: 24/3/2004
  + Version 2.0: 2006
  + Version 3.0: 2009
  + Version 4.0: 2013
  + Version 5.0: 2017
* Ứng dụng:
  + Mobile
  + Web
  + …
* Cấu trúc của Spring



* + Data Access / Integration

Nhóm này bao gồm JDBC, ORM, OXM, JMS và module Transaction. Những module này cung cấp khả năng giao tiếp với database

* + AOP, Aspects and Instrumentation

Những module này hỗ trợ cài đặt lập trình hướng khía cạnh (Aspect Oriented Programming), hỗ trợ tích hợp với AspectJ

* + Spring Core Container

Bao gồm các module spring core, beans, context và expression languate (EL).

Spring core, bean cung cấp tính năng IOC và Dependency Injection.

Spring Context hỗ trợ đa ngôn ngữ (internationalization), các tính năng Java EE như EJB, JMX.

Expression Language được mở rộng từ Expresion Language trong JSP. Nó cung cấp hỗ trợ việc setting/getting giá trị, các method cải tiến cho phép truy cập collections, index, các toán tử logic…

* + Web

Hay còn gọi là Spring MVC Nhóm này gồm Web, Web-Servlet… hỗ trợ việc tạo ứng dụng web.

* + Test

Tầng này cung cấp khả năng hỗ trợ kiểm thử với JUnit và TestNG.

1. **Tổng quan về Spring Boot**

* Spring boot là mã nguồn mở được phát triển bởi Pivotal Team và là 1 phần của hệ sinh thái Spring Framework.
* Spring boot hỗ trợ cho lập trình viên có thể dễ dàng tạo ra một sản phẩm spring applications mà không cần phải thực hiện thiết lập quá nhiều.
* Spring boot có khả năng tự động config gần như đầy đủ và người lập trình không cần phải tự làm.

1. **Spring Bean**

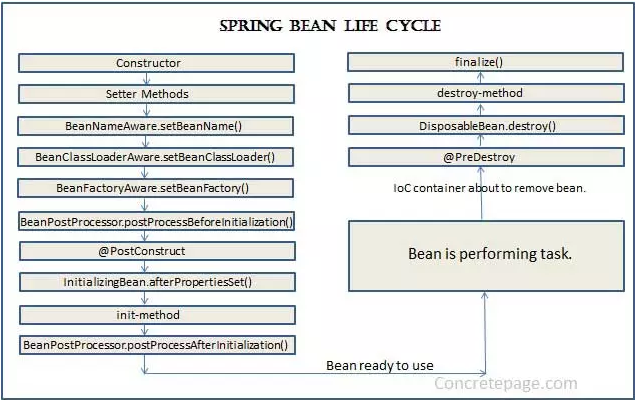
* Trong Spring, các đối tượng là các thành phần cốt lõi của ứng dụng và được quản lí bởi Spring IOC container được gọi là bean.
* Bean sẽ được khởi tạo, quản lí bởi IOC container.
* Bean sẽ được tạo ra bằng cách mà lập trình viên cung cấp cho container.
* Về IOC :
  + Là một design pattern được tạo ra để tuân thủ nguyên tắc dependence injection
  + Cung cấp cơ chế đơn giản để chứa và quản lí các đối tượng phụ thuộc.
  + Một object bị phụ thuộc sẽ cần 1 số lượng nhất định các object phụ thuộc được quản lí bởi IOC container.

🡺 IOC container sẽ cung cấp cách để object bị phụ thuộc có thể tương tác được với các object phụ thuộc.

* Bean Configulation.
  + Lập trình viên có thể thực hiện cung cấp cách cho container khởi tạo 1 bean cho 1 đội tượng nào đó.
  + @Configulation : đánh dấu 1 class được sử dụng để tạo Bean.
  + @ComponentScan: khai báo package để Spring có thể scan và inject Bean vào
  + @Autowired: Báo với Spring rằng Spring cần tìm bean thích hợp và inject vào.

1. **Lifecycle trong Spring**

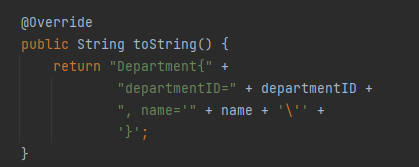
* Từ thời điểm được tạo ra cho đến khi bị phá đi, 1 vòng đời của bean sẽ được biểu diễn như sau:

****

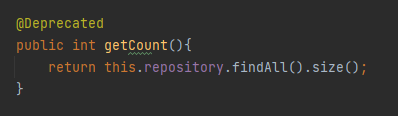
* + Khi IOC containner (ApplicationContext) tìm thấy bean cần quản lí, nó sẽ khởi tạo bằng contructor.
  + Thực hiện inject dependence vào bean bằng setter, và thực hiện các quá trình cài đặt khác cho bean như setBeanName, setBeanClassLoader, setBeanFactory, …
  + Hàm đánh dấu @PostContruct được gọi. (@PostContruct sẽ được đánh dấu cho 1 method duy nhất của bean. Method được đánh dấu đó sẽ được gọi sau khi bean được tạo ra và quản lý).
  + Tiền xử lý sau khi @PostContruct được gọi.
  + Bean sẵn sàng để sử dụng
  + Bean được sử dụng
  + Đến khi IOC không quản lí Bean nữa hoặc bị tắt đi thì sẽ gọi hàm được đánh dấu @PreDestroy trong bean (@PreDestroy sẽ được đánh dấu cho 1 method duy nhất của bean. Method được đánh dấu đó sẽ được gọi trước khi Bean bị xóa).
  + Xóa Bean.

1. **Annotations.**

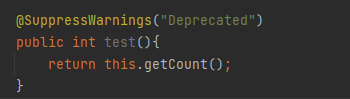
* Là một dạng meta data.
* Được đánh dấu vào class, interface, methods hoặc cả các fields để biểu diễn cho các thông tin thêm.
* Annotations được sử dụng thay thế cho các config XML.
* Annotations được sử dụng rất nhiều trong 1 project java, nó giúp cho việc viết code trở nên dễ dàng và ngắn gọn hơn.
* Annotations có rất nhiều loại nhưng chủ yếu nằm trong 2 phần chính: Build In (system hoặc framework) và custom (user define : thường được sử dụng cho quá trình validation).
* Build In: 1 số loại thường gặp như
  + @Override: đánh dấu vào method để báo rằng đây là 1 hàm được ghi đè ở class con.

****

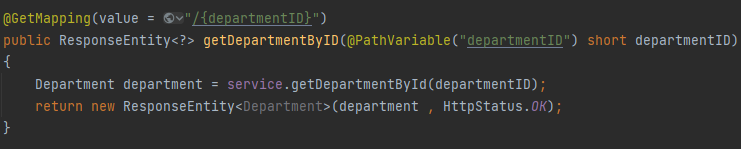
* + @Deprecatied: đánh dấu đây là 1 hàm đã cũ, và đã có các version cao hơn, java sẽ báo các hàm khác để có thể sử dụng

****

* + @SuppressWarrings : tắt đi 1 cảnh báo nào đó về code.

****

* Một số annotations thường được sử dụng đối với framework spring boot.
  + @SpringBootApplication : đánh đấu main của project spring boot
  + @RestController : đánh dấu 1 source api
  + @RequestMapping(value = “…"): config các cấu hình đường dẫn đến với source api
  + @GetMapping : khai báo function sử dụng phương thứ GET, có thể thực truyền vào đường dẫn, các tham số khác…
  + @PostMapping : khai báo function sử dụng phương thứ POST, có thể thực truyền vào đường dẫn, các tham số khác…
  + @PUT : khai báo function sử dụng phương thứ UPDATE, có thể thực truyền vào đường dẫn, các tham số khác…
  + @Delete : khai báo function sử dụng phương thứ DELETE, có thể thực truyền vào đường dẫn, các tham số khác…
  + @PathVariable : thực hiện truyền data cho phương thức.
  + @Param : có công dụng giống với @PathVariable

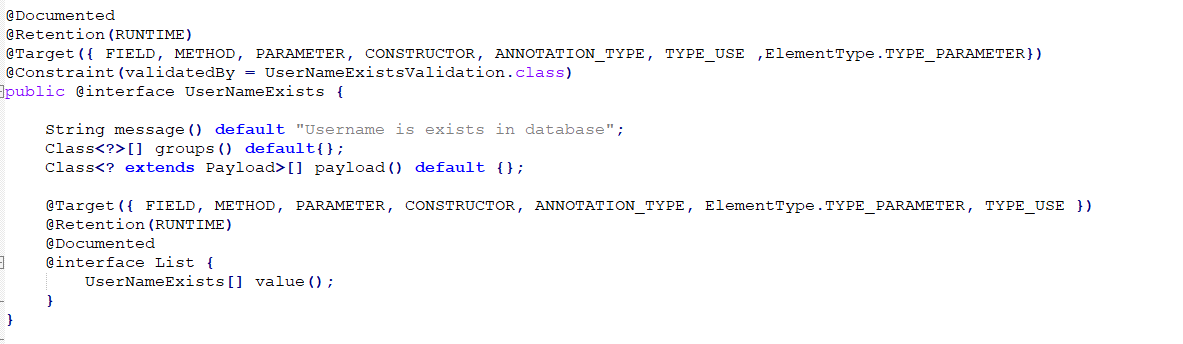


* + @RequestBody : dữ liệu đầu vào là dạng JSON
  + @CrossOrigin : thực hiện cho phép 1 server nào đó có thể truy cập vào source API.
  + @Service , @Repository , … khai báo là các thành phần của spring và yêu cầu spring tạo bean cho nó.
  + @Scope : khai báo về vòng đời và phạm vi mà 1 đối tượng được tạo ra (SingleTon , Request , Prototype,…).
  + @Autowired, @Resource , @Inject : thực hiện yêu cầu spring tạo bean thích hợp và inject vào (có thể thực hiện config cho field hoặc setter).
  + @Component : Đánh dấu 1 class là thành phần của spring để get Bean
  + @Bean: khai báo cách thức để spring tạo bean.
  + @Query, @NameQuery, @NameNativeQuery : thực hiện định nghĩa câu lệnh truy vấn cho function , class.
  + …

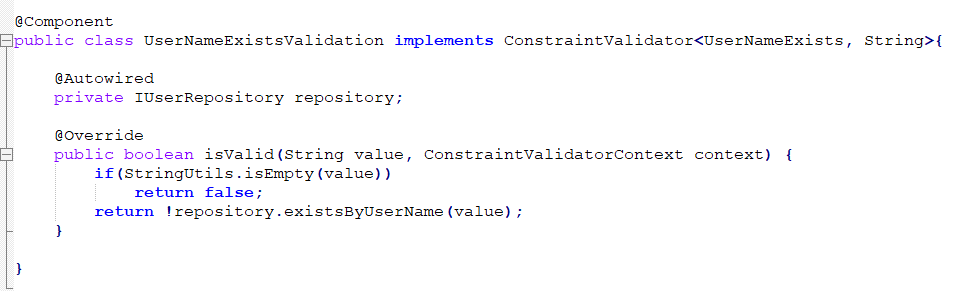
Ngoài ra còn có các annotation được sử dụng để thực hiện validation:

* @NotNull : giá trị không được null
* @Max,@Min : giá trị cần thỏa mã giới hạn
* @NotBlank : không được null và không được để khoảng trắng
* @Email : cần đúng định dạng email
* …
* Custom Annotation
  + Coder có thể thực hiện tạo annotation để sử dụng

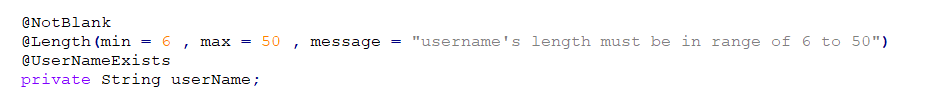
Vd: Tạo annotation để validation username có tồn tại trong db chưa.



UserNameExists.java



UserNameExistsValidation.java



UserDto.java