|  |  |
| --- | --- |
| Java Basic |  |
| Toán tử instanceof | sử dụng để kiểm tra một đối tượng có phải là thể hiển của một kiểu dữ liệu cụ thể không (lớp, lớp con, interface). |
| abstract | được sử dụng để khai báo một phương thức dạng abstract. Một phương thức abstract không có thân phương thức. |
| Upcasting | biến tham chiếu của lớp cha tham chiếu tới đối tượng của lớp con |
| **Polymorphism** | là một khái niệm mà chúng ta có thể thực hiện một hành động bằng nhiều cách khác nhau |
| **Nạp chồng phương thức** | trong java xảy ra nếu một lớp có nhiều phương thức có tên giống nhau nhưng khác nhau về kiểu dữ liệu hoặc số lượng các tham số |
| **Đa hình lúc runtime** | là quá trình gọi phương thức đã được ghi đè trong thời gian thực thi chương trình |
| Tính trừu tượng | là một tiến trình ẩn các cài đặt chi tiết và chỉ hiển thị tính năng tới người dùng. |
| Từ khóa super | là một biến tham chiếu được sử dụng để tham chiếu trực tiếp đến đối tượng của lớp cha gần nhất |
| Interface | là một kỹ thuật để thu được tính trừu tượng hoàn toàn và đa kế thừa trong Java. |
|  |  |
| @Component | Là annotation đánh dấu trên class giúp Spring biết nó là Bean |
| @Autowired | Là annotation biểu thị rằng dependency sẽ được tự động inject vào object. |
| @Scope  Scope singleton (mặc định) | Container tạo ra một instance mới |
| @Primary | Thêm vào class ứng với Bean cần ưu tiên. |
| @Qualifier | @Qualifier("ten\_cua\_bean\_muon\_inject") đi kèm với @Autowired |
| @Service | xử lý các nghiệp vụ của ứng dụng. |
| **Bean** | là các Java Object tạo nên khung sườn của một application. |
| @Bean | là một Annotation được đánh dấu trên các method cho phép Spring Boot biết được đây là Bean và sẽ thực hiện đưa Bean này vào Context. |
| @Configuration | là một Annotation đánh dấu trên một Class cho phép Spring Boot biết được đây là nơi định nghĩa ra các Bean. |
| @RequestParam | | để binding dữ liệu từ query parameter vào tham số của method |
| @RestController | | trả về dữ liệu dưới dạng JSON. |
| @ModelAttribute | | Spring Boot sẽ binding dữ liệu từ form request vào param |
| @Id | | đánh dấu khóa chính |
| @Entity | | đánh dấu class này là Bean loại entity |
| @GeneratedValue(strategy = ...) | | cấu hình tự động gen giá trị cho khóa chính mỗi khi insert một bản ghi mới |
| @RequestBody | | chuyển chuỗi thông tin client gửi lên(request) dưới dạng JSON thành một object |
| GET (SELECT): | | Trả về một Resource hoặc một danh sách Resource. |
| POST (CREATE): | | Tạo mới một Resource. |
| PUT (UPDATE) | | Cập nhật thông tin cho Resource. |
| DELETE (DELETE): | | Xoá một Resource. |
| API (Application Programming Interface) | | là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. |
| REST (REpresentational State Transfer) | | là một dạng chuyển đổi cấu trúc dữ liệu |
| **tight-coupling** hay "liên kết ràng buộc" | | là một khái niệm trong Java ám chỉ việc mối quan hệ giữa các Class quá chặt chẽ. |
| **loosely-coupled** | | là cách ám chỉ việc làm giảm bớt sự phụ thuộc giữa các Class với nhau. |
| **Dependency Injection (DI)** | | (nôm na) là một design pattern giúp **loosely coupled(ràng buộc lỏng lẻo)** |
| **Inversion of control (IOC)** | | Định nghĩa toàn bộ dependency có trong project, đưa vào một cái kho. Bất kỳ Class nào khi khởi tạo, nó cần dependency gì, thì cái framework đang sử dụng sẽ tự tìm trong kho rồi inject vào đối tượng |
| **Swagger** | | là 1 open source dùng để phát triển, thiết kế, xây dựng và làm tài liệu cho các hệ thống RESTfull Web Service. |
|  | |  |
|  | |  |
| 1xx | | hàm ý mang thông tin. |
| 2xx | | hàm ý thành công |
| 3xx | | hàm ý điều hướng |
| 4xx | | hàm ý là có lỗi từ phía client side |
| 5xx | | hàm ý là có lỗi phía máy chủ (server) |
| **200 OK**  Phản hồi tiêu chuẩn cho các yêu cầu HTTP thành công | | Standard response for successful HTTP requests |
| 404 not found  Không thể tìm thấy tài nguyên được yêu cầu nhưng có thể có trong tương lai. Các yêu cầu tiếp theo của khách hàng được cho phép. | | The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible. |
| **502  Bad Gateway**  Máy chủ đang hoạt động như một cổng hoặc proxy và nhận được phản hồi không hợp lệ từ máy chủ ngược dòng | | The server was acting as a [gateway](https://en.wikipedia.org/wiki/Gateway_(telecommunications)) or proxy and received an invalid response from the upstream server |
| **504 Gateway Timeout**  Máy chủ đang hoạt động như một cổng hoặc proxy và không nhận được phản hồi kịp thời từ máy chủ ngược dòng | | The server was acting as a gateway or proxy and did not receive a timely response from the upstream server |
| Thymeleaf Standard | |  |
| ${...} | | Giá trị của một biến. |
| \*{...} | | Giá trị của một biến được chỉ định |
| #{...} | | Lấy message |
| @{...} | | Lấy đường dẫn URL dựa theo context của server |
|  | |  |
|  | |  |
| **Basic Object** | | **Các đối tượng cơ bản** |
| #locale | | Một đối tượng cung cấp các thông tin liên quan đến **Locale** (Địa phương) |
| #request | | (Chỉ trong môi trường **Web**) Đối tượng **HttpServletRequest** |
| #response | | (Chỉ trong môi trường **Web**) Đối tượng **HttpServletResponse** |
| #session | | (Chỉ trong môi trường **Web**) Đối tượng **HttpSession** |
| #servletContext | | (Chỉ trong môi trường **Web**) Đối tượng **ServletContext** |
|  | |  |
| ****Utility Objects**** | |  |
| #execInfo | | Thông tin về **Template** đang được xử lý. |
| #messages | | Các phương thức để làm việc với các **message**. |
| #uris | | Các phương thức để **escape** các phần của **URLs/URIs**. |
| #conversions | | Các phương thức để thực thi "dịch vụ chuyển đổi" (**conversion service**) đã được cấu hình (Nếu có). |
| #dates | | Các phương thức để định dạng đối tượng **java.util.Date**, hoặc lấy các thông tin liên quan như ngày, tháng, năm,.. |
| #calendars | | Giống với **#dates**, nhưng với đối tượng **java.util.Calendar**. |
| #numbers | | Các phương thức để định dạng các đối tượng số (Number). |
| #strings | | Các phương thức cho các đối tượng **String**. Chẳng hạn **contains, startsWith**, .. |
| #objects | | Các phương thức cho các đối tượng nói chung. |
| #bools | | Các phương thức cho việc đánh giá **boolean**. |
| #arrays | | Các phương thức cho các mảng (array). |
| #lists | | Các phương thức cho **lists**. |
| #sets | | Các phương thức cho **sets**. |
| #maps | | Các phương thức cho maps |
| #aggregates | | Các phương thức để tính tổng, giá trị trung bình,.. trên một tập hợp (collection) hoặc mảng (array). |
|  | |  |
| JDBC (Java Database Connectivity) | | là một java API được sử dụng để kết nối và thực hiện truy vấn với cơ sở dữ liệu |
| FetchType.LAZY | | Khi bạn find, select một đối tượng từ database thì JPA sẽ không lấy ra các các đối tượng phụ thuộc, chỉ lấy ra các thông tin có trong bảng ứng với entity. |
| FetchType.EAGER | | Khi bạn find, select một đối tượng từ database thì JPA sẽ lấy ra luôn ra cả các đối tượng liên quan phụ thuộc. |
| Authentication | | 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 của bạn. |
| Authorization | | là tiến trình quyết định xem một principal có được phép thực hiện một hành động trong ứng dụng của bạn hay không. Trước khi diễn tiến tới Authorization, principal cần phải được thiết lập bởi Authentication. |
| **SecurityContext** | | là interface cốt lõi của Spring Secure, lưu trữ tất cả các chi tiết liên quan đến bảo mật trong ứng dụng |
|  | |  |
| JUnit5 | |  |
| **@Test** | | Định nghĩa một method là test method |
| **@RepeatedTest(n)** | | Lặp lại việc kiểm thử n lần |
| **@TestFactory** | | Đánh dấu method là một Factory cho dynamic test. |
| **@BeforeEach** | | Thực thi trước khi test. Cái này thường được đánh dấu những method cho việc chuẩn bị môi trường |
| **@AfterEach** | | Thực thi sau khi test. Thường được đánh dấu những method có nhiệm vụ dọn dẹp môi trường |
| **@BeforeAll** | | Thực hiện một lần duy nhất và là method chạy đầu tiên chạy một class test. |
| **@AfterAll** | | Thực hiện một lần duy nhất và là method cuối cùng chạy trong một class test. |
| **@Neste** | | Cho phép bạn lồng các class test để thực hiện theo thứ tự. |
| **@Tag(“{tagName}”)** | | Việc kiểm thử trong JUnit 5 có thể được lọc bởi tag. Ví dụ như bạn chỉ muốn chạy các test case được đánh dấu là “run” thì sẽ đặt annotation @Tag(“run”). |
| **@ExtendWith** | | Cho phép bạn đăng ký một lớp Extension để tích hợp với một hoặc nhiều điểm mở rộng. |
| **@Disabled** hoặc **@Disabled(“Why disabled”)** | | Đánh dấu những method test nào có thể bỏ qua không cần run. |
| **DisplayName(“{name}”)** | | giúp hiển thị tên các method khi chạy một cách dễ nhìn và dễ đọc hơn. |
|  | |  |
| Kafka | |  |
| Kafka | | Kafka là **hệ thống message pub/sub** phân tán (distributed messaging system) có khả năng scale rất tốt |
| **Message** | | Thông tin được gửi đi, có thể là text, binary, Json hoặc một định dạng format nào đó. |
| **Broker** | | Một host có thể chạy nhiều server kafka, mỗi server như vậy gọi là một broker. Các broker này cùng trỏ tới chung 1 zookeeper, gọi là cụm broker (hay là Clusters). |
| **Topics** | | Message được lưu và phân loại trên đĩa cứng theo topics. Nhìn về mặt database thì topic giống như một table trong cơ sở dữ liệu quan hệ, và mỗi message như một bản ghi của table đó |
| **Producer** | | Chương trình/service tạo ra message, đẩy message publish vào Topic. |
| **Consumer** | | Chương trình/service có chức năng subscribe vào một Topic để tiêu thụ, xử lý các message đó. |
| Key words:  - Loosed couple: hiểu biết về nhau ít  - DI: dấu hiệu: nhiều lựa chọn cho 1 tính năng, tính năng có thể thay đổi  Cách thực hiện: “găm” tool quanh người, chọn tool khi cần  Danh từ - abstract, động từ - interface  - Exception: cấu trúc giống if else,  Truy ngược để tìm nguồn gốc lỗi có thể xảy ra  Rethrow nếu ko thể xử lý  Finally - try with  - phân tích đề: danh từ, trợ danh từ, động từ  Key words:  - danh từ, trợ danh từ, động từ  - tên hàm khớp với nội dung hàm  - mỗi hàm chỉ làm 1 việc  - cơ chế 1 cửa  - refine code  Kết lại:  - tuân thủ nguyên tắc: tên hàm phải khớp với nội dung hàm  - tuần thủ nguyên tắc: 1 hàm 1 tính năng  - tuần thủ nguyên tắc: 1 cửa, viết 1, dùng nhiều, sửa 1  - chú ý scope khi lập trình: hàm nào cho phép kế thừa, hàm nào ko, hàm nào cho dùng bên ngoài, hàm nào ko | | |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |