Lập trình hướng đối tượng có 4 tính chất:

-Trừu tượng

-Đa hình

-Kế thừa

-Đóng gói

Tính trừu tượng là từ mô tả của chương trình tìm ra các đặc trưng,hành động để trừu tượng hóa các đối tượng các class

-Tính đóng gói là đóng gói các biến,method thành các class,các class thành 1 package

-Tính đóng gói giúp che dấu thông tin và nhưng xử lý bên trong của đối tượng

-Các đối tượng khác không thể tác động trực tiếp đến dữ liệu bên trong và làm thay đổi trạng thái của đối tượng mà bắt buộc phải thông qua các phương thức công khai do đối tượng đó cung cấp

-Tính chất này giúp tang tính bảo mật của đối tượng và tránh tình trạng dữ liệu bị hỏng ngoài ý muốn

Tính kế thừa là sử dụng các thuộc tính,method có sẵn từ các class khác mà không cần xây dựng lại từ đầu

-Tính đa hình cho phép các đối tượng khác nhau thực thi chức năng giống nhau theo những cách khác nhau

-Tính đa hình bao gồm orverloading (nạp chồng),orverriding(ghì đè)

Orverloading là sử dụng các method có tên giống nhau nhưng khác tham số đầu vào khác nhau

Orveriding là viết lại,định nghĩa lại method có sẵn mà nó kế thừa từ lớp cha

2. Getter/Setter

Là phương thức lấy dữ liệu hoặc chỉnh sửa thông tin cho dữ liệu của đối tượng

3. Private : truy cập trong nội bộ lớp

Default : Truy cập trong nội bộ package

Protected : Thành phần được bảo vệ,bị hạn chế truy cập từ bên ngoài

Public : Thành phần công khai,tự do truy cập từ bên ngoài

**Từ khóa static trong Java** được sử dụng chính để quản lý bộ nhớ.

Spring

**Annotation**trong **Spring Boot** là một dạng siêu dữ liệu cung cấp dữ liệu về một program. Nói cách khác, annotation được sử dụng để cung cấp thông tin **bổ sung** về một program. Các annotation không phải là một phần của ứng dụng mà chúng ta phát triển. Nó không có ảnh hưởng trực tiếp đến hoạt động của code mà họ chú thích. Nó không thay đổi hành động của chương trình đã biên dịch.

**@Required:** nó áp dụng cho method bean setter. Nó chỉ ra rằng bean được chú thích phải được điền vào thời điểm cấu hình với thuộc tính bắt buộc, nếu không nó sẽ ném một ngoại lệ **BeanInitizationException**.

**@Autowired**: biểu thị rằng các thuộc tính sẽ được auto wired. Nó được sử dụng để autowire Spring Bean trên các method setter, instance variable và contructor. Khi chúng ta sử dụng @Autowired, Spring Container auto wire bean bằng cách match data-type.

**@Configuration**: là một annotation của class. Class được đánh dấu annotaiton này được Spring Container sử dụng làm nguồn định nghĩa bean.

**@ComponentScan**: là annotation khai báo ở cấp độ class, giống như cái tên của nó, annotation **@ComponentScan** dùng để scan toàn bộ bean trong ứng dụng, mặc định nó sẽ scan tất cả bean trong package ở vị trí đặt class chưa hàm main, ngoài ra chúng ta cũng có thể tùy chỉnh package cần scan. thường thì annotation **@ComponentScan** sẽ đi chung vớ **@ComponentScan**.

Các param trong @ComponentScan:

* **basePackages**: dùng để chỉ ra những package cần scan, bạn có thể truyền vào một hoặc nhiều package name
* **basePackageClasses**: chỉ ra những class cần scan
* **excludeFilters**: dùng để filter loại trừ những bean or package không cần scan
* **includeFilters**: Dùng dể filter những bean or package cần scan.
* **lazyInit**: dùng để lazy load bean, default = false
* **resourcePattern**: chỉ ra các pattern file để Spring có thể scan, default pattern "\*\*/\*.class"

**@Bean**: nó là một annotation cho method. Nó là sự thay thế của thẻ XML <bean>. Nó cho biết method tạo ra một bean được Spring Container quản lý.

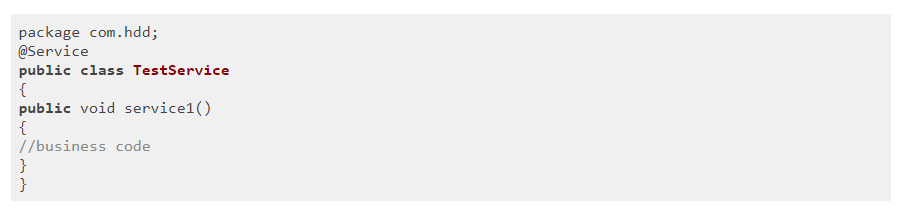
**Spring Framework Stereotype Annotations**

**@Component:** là một annotation của class. Nó dùng để đánh dấu class **Java**là một bean. Một class **Java**được đánh dấu **@Component** được tìm thấy trong classpath. Spring Framework chọn nó và cấu hình trong ngữ cảnh ứng dụng như một Spring Bean.

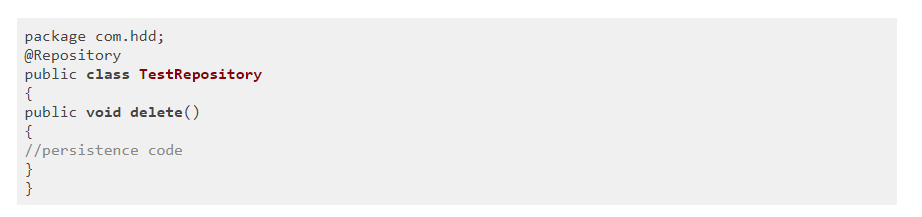
**@Controller**: **@Controller** là một annotation ở class. Là một chuyên môn hóa của **@Component**. Nó đánh dấu class là một class để xử lý request web. Nó thường được sử dụng để phục vụ các request từ UI. Mặc định thì nó trả về một chuỗi cho biết route nào cần redirect. Nó chủ yếu được sử dụng với annotation **@RequestMapping**.



**@Service:** nố thường được sử dụng ở class. Nó cho biết Spring biết rằng class này chứa **business login**.



**@Repository:** nó thường được dụng ở class. Repository là một **DAOs** (Data Access Object) cái mà sẽ truy cập trực tiếp vào database. Repository thực hiện tất cả các hoạt động liên quan đến database.



**Spring Boot Annotations**

* **@EnableAutoConfiguration:** Nó tự động cấu hình bean có trong classpath và cấu hình nó để chạy các phương thức. Việc sử dụng chú thích này bị giảm trong bản phát hành Spring Boot 1.2.0 vì các nhà phát triển đã cung cấp một giải pháp thay thế cho chú thích, tức là **@SpringBootApplication**.
* **@SpringBootApplication:** Nó là sự kết hợp của ba chú thích **@EnableAutoConfiguration**, **@ComponentScan** và **@Configuration**.

**Spring MVC and REST Annotations**

* **@RequestMapping:** nó được sử dụng để map các request. Nó có nhiều phần tử tùy chọn như **consumes, header, method, name, params, path, produces**, và **value**. Chúng tôi sử dụng nó với lớp cũng như phương thức.
* **@GetMapping**: nó map HTTP GET request trên method cụ thể. Nó dùng để tạo một điểm cuối trong web service
* **@PostMapping**: sử dụng khi gửi data tới server để tạo ra tài nguyên mới.

**@PutMapping**: sử dụng để thay đổi thông tin của một tài nguyên đã tồn tại trong hệ thống bằng cách sử dụng nội dung trong body của request.

**@DeleteMapping**: sử dụng để xóa một resource chỉ định.

**@PatchMapping**: Sử dụng để áp dụng sửa đổi 1 phần cho resource.

* **@RequestBody**: Nó được sử dụng để liên kết yêu cầu **HTTP**với một đối tượng trong một tham số phương thức. Bên trong nó sử dụng **HTTP MessageConverters**để chuyển đổi phần thân của yêu cầu. Khi chúng tôi chú thích một tham số phương thức với @RequestBody, **Spring**sẽ liên kết phần body yêu cầu HTTP đến với tham số đó.
* **@RequestHeader**: Nó được sử dụng để lấy thông tin chi tiết về tiêu đề yêu cầu HTTP. Chúng tôi sử dụng chú thích này như một tham số phương thức. Các phần tử tùy chọn của chú thích là **name, required, value, defaultValue**. Đối với mỗi chi tiết trong tiêu đề, chúng ta nên chỉ định các chú thích riêng biệt. Chúng ta có thể sử dụng nó nhiều lần trong một phương pháp.
* **@RestController**: @**RestController là** một composed annotation được kết từ annotation @Controller và **@ResponseBody**, khi đặt một annotation @**RestController** trên một class controller thì mọi method controller trong class đó sẽ được thừa hưởng annotation @ResponseBody và response data trong controller này sẽ được trả về dưới dạng message.
* **@RequestAttribute**: Nó liên kết một tham số phương thức với thuộc tính yêu cầu. Nó cung cấp quyền truy cập thuận tiện vào các thuộc tính yêu cầu từ một phương thức bộ điều khiển. Với sự trợ giúp của chú thích **@RequestAttribute**, chúng tôi có thể truy cập các đối tượng được điền ở phía máy chủ.