Một class không nên phụ thuộc và class kế thừa cấp thấp mà chỉ nên phụ thuộc và lớp Abstraction hoặc Interface.

Khắc phục:

Graphical user interface, text, application, email

Description automatically generated

**Inversion Control:**

**Lombok:**

Ko cần getter và setter:

Graphical user interface, text, application

Description automatically generated

* **controller**: các bean tiếp nhận request và trả về response cho client.
* **repository**: các bean thao tác trực tiếp với cơ sở dữ liệu.
* **service**: các bean xử lý logic, trung gian giữa repository và controller
* **resources**: các tài nguyên tĩnh như hình ảnh, css, javascript.
* Khi thêm **@Component** trên clas, ta đã đánh dấu class này là một Bean.
* Khi chạy chương trình, Spring sẽ dò tìm toàn bộ các class cùng package hoặc ở trong các package cùng cấp với class Application. Các class được đánh dấu sẽ được khởi tạo bằng contructor không tham số và tạo ra một instance rồi đưa vào **ApplicationContext** quản lý. ApplicationContext ở đây chính là **IoC Container**.

Diagram

Description automatically generated

* **@Controller** là một annotation đánh dấu một class là **@Component** với ý nghĩa đây là Bean đón nhận các **Request** và trả về **Response** cho client. Bean này thường sẽ sử dụng các **Service** là dependency hoặc có thể sử dụng trực tiếp các **Bean Repository**
* **@RequestMapping** được đặt trên **phương thức** sẽ xử lý request hoặc trên **class** Controller để xác định đường dẫn request sẽ được xử lý. Tạo Controller:

Graphical user interface, text, application, email

Description automatically generated

* Khi muốn trả về nội dung một cách trực tiếp ta sẽ thêm annotation @ResponseBody trên phương thức hoặc trên class Controller nếu muốn áp dụng cho toàn bộ các phương thức nội bộ
* Dữ liệu trả về với @ResponseBody sẽ có dạng **JSON**
* Nếu muốn tạo một Controller dạng Restful API thì nên sử dụng **@RestController**. Đây cũng là một @Controller nhưng được bổ sung thêm **@ResponseBody**
* **@GetMapping** sẽ thay thế cho @RequestMapping với method **GET**, điều này cũng áp dụng cho các method khác.

Quy tắc đặt tên cho Restful API: eg: api/public/student

* **api:** là tiền tố đánh dấu đây là một api để phân biệt với các loại URI khác.
* **public:** phân quyền ai có thể sử dụng api này, có thể là: public, user, admin.
* **students:** nội dung của api này, thêm hậu tố số nhiều nếu kết quả trả về dạng số nhiều

Swagger UI:

Maven:

<dependency>  
 <groupId>io.springfox</groupId>  
 <artifactId>springfox-swagger2</artifactId>  
 <version>2.8.0</version>  
</dependency>

<dependency>  
 <groupId>io.springfox</groupId>  
 <artifactId>springfox-swagger-ui</artifactId>  
 <version>2.8.0</version>  
</dependency>

Cấu hình trong main:

@EnableSwagger2

….

@Bean

public Docket api() {

return new Docket(DocumentationType.SWAGGER\_2)

.select()

.apis(RequestHandlerSelectors.any())

.paths(PathSelectors.ant("api/\*\*"))

.build();

}