[Introduction to Project Lombok | Baeldung](https://www.baeldung.com/intro-to-project-lombok)

@Data: @getter + @setter (no constructor)

// on property

*@Getter(lazy = true): Cache data cho property, app sẽ lấy data đã được cached (đã load trước đó) cho mỗi lần gọi getter của property đó,*  ***chỉ load lại data khi mà giá trị của TT = null. => thường dùng cho các data được load từ file config/ bảng config mà giá trị ít khi thay đổi.***

*@Setter(AccessLevel.Protected): phương thức setter sẽ chỉ được nhìn thấy bên trong 1 package, AccessLevel.private sẽ khong thể setid từ bên ngoài*

***@Tostring:*** *define toString for us*

***@Tostring.Exclude:*** *on a property to not add the property info to ToString method or @ToString(exclude = {“field,…”}) has the same effect on muilt fields.*

***@EqualsAndHashCode(of = {“id, fieldname…”}):*** *gen equals and hashcode for us* ***using fields in {}***

***=> @EqualsAndHashCode should not use for entity***

***@EqualsAndHashCode.EXCLUDE*** *: on property to abandon that property from equals and hashcode or we can use* ***@EqualsAndHashCode(exclude = {“fields,.. })*** *has the same effect on muilt fields*

***// {***

***@Builder: on classes***

***@Accessor(fluent = true)***

*Mỗi class có 1 builder ex: Student student = Student.Buider()*

*.name(“quan”) // setter*

*.age(10) // setter*

*.build();*

*Sout(student.name()) // getter*

***// }***

*@RequireArgsConstructor: on class*

*+* ***Sinh ra constructor*** *cho các thuộc tính là final, @nonnull mà chưa được gán các giá trị, các giá trị truyền vào bắt buộc phải có giá trị, nếu null sẽ throw ex.*

*+* ***Khi sinh constructor sẽ không chứa các thuộc tính sau****: không phải final, final nhưng đã gán giá trị, thuộc tính static, thuộc tính @non-null nhưng đã gán giá trị.*

* *Khi kết hợp @RequireArgConstructor + khai báo final cho các thuộc tính =>* ***sẽ thay thế được cho autowired***
* @RestController  
  @RequestMapping("/api")  
  @Tag(name = "People resources-quanzip")  
  @RequiredArgsConstructor  
  public class PeopleResource {  
    
   private final PeopleRepository peopleRepository;  
   private final PeopleMapper peopleMapper;  
    
   @GetMapping(value = "/people")  
   @Operation(description = "Xem danh sách mọi người trong hệ thống")  
   @Schema(description = "D/s mọi người trong hệ thống")  
   @ApiResponses({  
   @ApiResponse(responseCode = "200", description = "OK"),  
   @ApiResponse(responseCode = "404", description = "Page Not found"),  
   @ApiResponse(responseCode = "403", description = "No permission!"),  
   })  
   public List<PeopleDTO> getALlPeople(){  
   return peopleMapper.toDto(peopleRepository.findAll());  
   }  
  }
* ***Nếu chỉ sử dụng @AllArgsConstructor: các tham số sẽ là tất cả các thuộc tính trong class.***

***@SneakyThrows: thay thế cho các checked Exception nhưng ta chắc chắn sẽ không xảy ra.***

@SneakyThrows

**public** String **resourceAsString**() {

**try** (**InputStream** is = this.getClass().getResourceAsStream("sure\_in\_my\_jar.txt")) {

**BufferedReader** br = **new** **BufferedReader**(**new** **InputStreamReader**(is, "UTF-8")); **return** br.lines().collect(Collectors.joining("\n"));

}

}

***@Cleanup: Thay thế cho việc (close các tài nguyên như In/outStream, writer,… mannualy hoặc ấu trúc try-with-resources).***

@Cleanup **InputStream** is = this.getClass().getResourceAsStream("res.txt");

**@Log or @Slf4j**: for using log.info/error without declaring instance.

**@Synchronize** : better than **synchronized of java**

**Lombok sẽ** tạo ra 1 $lock cho phương thức thường và $LOCK cho static method.