1. **Các bước tạo project spring boot web :**

**B1**: Tạo project maven

**B2**: Thêm dependency spring-boot-starter-web

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<version>2.1.1.RELEASE</version>

</dependency>

spring-boot-starter-web đã bao gồm spring, spring-web, spring-webmvc, tomcat nên không cần add server tomcat ngoài cho project nữa.

**B3**: Tạo class main như dưới:

@SpringBootApplication

**public** **class** SpringBootRestApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringBootRestApplication.**class**, args);

}

}

Annotation @SpringBootApplication tương đương với việc dùng kết hợp cả 3 annotation @Configuration, @EnableAutoConfiguration, @ComponentScan. Mặc định nó sẽ scan các component trong cùng package và ở các package con.

1. **Tạo các controller**

* Controller là class dùng để tiếp nhận request và trả về response cho client

@RestController

**public** **class** HelloWorld {

@RequestMapping("/")

**public** String hello() {

**return** "Hello World";

}

}

* Annotation @RestController đã gộp 2 annotation @Controller và @ResponseBody vào làm một. Annotation @ResponseBody tự động chuyển object của java thành dạng json trước khi trả về cho client => Content-Type: application/json. Nếu không có annotation này hệ thống sẽ báo lỗi.
* Khi muốn trả về status code cụ thể cho client, dùng annotation @ResponseStatus(HttpStatus.xxx)
* Annotation @RequestBody: gắn vào 1 tham số của method dùng để map dữ liệu người dùng gửi lên với Object của java.
* Annotation @RequestMapping…. method = RequestMethod.POST
* Annotation @PathVariable: dùng để lấy các tham số trên url

@RequestMapping("/hello/{id}")

**public** String helloPathVariable(@PathVariable("id") String id) {

**return** String.*format*("Hello World, your id is %s", id);

}

* ResponseEntity …

1. **Xử lý lỗi trong spring**
   1. Xử lý exception ở controller bằng annotation @ExceptionHandler

@PostMapping("/post-user2")

**public** String postUser2(@RequestBody User user) **throws** PasswordValidateException {

**if** (user.getPassWord().length() < 8) {

**throw** **new** PasswordValidateException("password must be at least 8 characters");

}

**return** user.toString();

}

@ExceptionHandler(PasswordValidateException.**class**)

**public** ResponseEntity<ErrorResponse> handleException(PasswordValidateException ex) {

List<String> details = **new** ArrayList<>();

details.add(ex.getLocalizedMessage());

ErrorResponse error = **new** ErrorResponse("Validation errors", details);

**return** **new** ResponseEntity<ErrorResponse>(error, HttpStatus.***NOT\_FOUND***);

}

* Chỉ có tác dụng trong controller đó
  1. Xử lý exception bằng annotation @ControllerAdvice (cấp độ global)

@ControllerAdvice

**public** **class** GlobalHandleException **extends** ResponseEntityExceptionHandler {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@ExceptionHandler(ValidateException.**class**)

**public** ResponseEntity<ErrorResponse> resourceNotFoundException(ValidateException ex) {

List<String> details = **new** ArrayList<>();

details.add(ex.getLocalizedMessage());

ErrorResponse errorDetails = **new** ErrorResponse("validation errors", details);

**return** **new** ResponseEntity<ErrorResponse>(errorDetails, HttpStatus.***BAD\_REQUEST***);

}

@Override

**protected** ResponseEntity<Object> handleMethodArgumentNotValid(

MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatus status,

WebRequest request) {

List<String> errors = **new** ArrayList<String>();

**for** (FieldError error : ex.getBindingResult().getFieldErrors()) {

errors.add(error.getField() + ": " + error.getDefaultMessage());

}

**for** (ObjectError error : ex.getBindingResult().getGlobalErrors()) {

errors.add(error.getObjectName() + ": " + error.getDefaultMessage());

}

ErrorResponse errorDetails = **new** ErrorResponse("validation errors", errors);

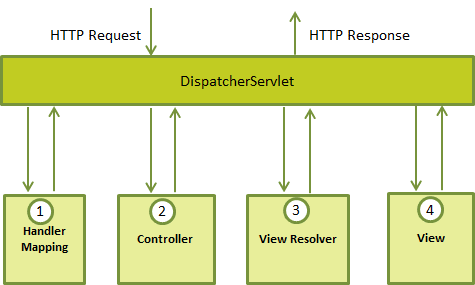
**return** **new** ResponseEntity<Object>(errorDetails, HttpStatus.***BAD\_REQUEST***);

}

}

**4. Debug Spring boot**

logging.level.org.springframework = DEBUG



**5. Validate trong spring**

- Dùng annotation @Valid với request body

- Một số validation annotation thông dụng (javax.validation): @NotBlank, @Size,

6. Filtering Field

- Khi muốn ẩn 1 field của class (bean) khi trả về cho người dùng, ta thêm annotation @JsonIgnore trước field đó.

@JsonIgnore

**private** String field3;

* Khi muốn ẩn field tùy chỉnh, các request khác nhau thì trả về các field khác nhau ta dùng annotation @JsonFilter

@JsonFilter("filterField")

**public** **class** DynamicFilterBean {}

@GetMapping("/filter/dynamic1")

**public** MappingJacksonValue dynamicFilter1() {

DynamicFilterBean bean = **new** DynamicFilterBean("field1", "field2", "field3");

SimpleBeanPropertyFilter filterField = SimpleBeanPropertyFilter.*filterOutAllExcept*("field3");

FilterProvider filters = **new** SimpleFilterProvider().addFilter("filterField", filterField);

MappingJacksonValue mappingJacksonValue = **new** MappingJacksonValue(bean);

mappingJacksonValue.setFilters(filters);

**return** mappingJacksonValue;

}

**6. Lấy các giá trị trong file application.properties**