**Proper Usage of Annotations in SpringBoot - 2025**

@GetMapping("/emp/{id}")

**public** ResponseEntity<Employee> getEmployeeById(@PathVariable **int** id) {

Employee emp = **new** Employee(id, "John", "Doe");

**return** ResponseEntity.*ok*().body(emp);

// return ResponseEntity.status(HttpStatus.OK).body(emp);

}

**@PathVariable(“id”) int.id 🡺 Not Required to write**

**@PathVariable int.id 🡺 Accepted and works fine**

**@RequestParam("firstName") String firstName 🡺 Not Required to write**

**@RequestParam String firstName 🡺 Accepted and works fine**

**@RequestHeader(“key”) String key 🡺 Not Required to write**

**@RequestHeader String key 🡺 Accepted and works fine**

**@ExceptionHandler(NotFoundException.class) 🡺 Not Required to write**

**@ExceptionHandler 🡺 Accepted and works fine**

**How to Use ResponseEntity Properly**

**return** ResponseEntity.*status*(HttpStatus.***OK***).body(emp);

**return ResponseEntity.ok().body(emp);**

you can also write like this.

**return** ResponseEntity

.*status*(HttpStatus.***OK***)

.header("Response-Header-1", "value-1")

.body(emp);

In case of **@ControllerAdvice** with **@ExceptionHandler**, you have to use in the following way.

**return** **ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(appError);**

A basic structure is given below

**return ResponseEntity**

**.*status*(HttpStatus.*OK*)**

**.header("Response-Header-1", "value-1")**

**.contentType(null)**

**.contentLength(0)**

**.body(emp);**

**How to properly use @ControllerAdvice with @ExceptionHandler**

**@ControllerAdvice**

**public** **class** GlobalExceptionHandler {

**@ExceptionHandler(NotFoundException.class)**

**public** ResponseEntity<AppError> handleNotFoundException(NotFoundException ex) {

AppError appError = **new** AppError(HttpStatus.***NOT\_FOUND***.value(), ex.getMessage());

**return** ResponseEntity.*status*(HttpStatus.***NOT\_FOUND***).body(appError);

}

**@ExceptionHandler**

**public** ResponseEntity<AppError> handleBadRequestException(BadRequestException ex) {

AppError appError = **new** AppError(HttpStatus.***BAD\_REQUEST***.value(), ex.getMessage());

**return** ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).body(appError);

}

}

**How to display an Image in the Browser**

@GetMapping("/display/image")

**public** **ResponseEntity<byte[]>** getImage() {

**byte**[] imageBuffer = {};

**try** {

imageBuffer = **Files.*readAllBytes***(Paths.*get*("C:/WS-2/data/apple.jpg"));

} **catch** (IOException e) {

e.printStackTrace();

}

**return** ResponseEntity

.*status*(HttpStatus.***OK***)

**.contentType(MediaType.*IMAGE\_JPEG*)**

.body(imageBuffer);

}

**.contentType(MediaType.*IMAGE\_JPEG*) 🡸 Important**

**How to download a file**

@GetMapping("/download/image")

**public** **ResponseEntity<byte[]>** downloadImage() {

**byte**[] imageBuffer = {};

String fileName = "fruit.jpg";

**try** {

imageBuffer = **Files.*readAllBytes***(Paths.*get*("C:/WS-2/data/apple.jpg"));

} **catch** (IOException e) {

e.printStackTrace();

}

**return** ResponseEntity

.*status*(HttpStatus.***OK***)

**.header("Content-Disposition", "attachment; filename=\"" + fileName + "\"")**

.body(imageBuffer);

}

**.header("Content-Disposition", "attachment; filename=\"" + fileName + "\"")** **🡸 Important**

**How to upload a file**

@PostMapping("/upload/file")

**public** ResponseEntity<String> uploadFile(@RequestParam("file") MultipartFile file) {

Path fileStorageLocation = Paths.*get*("C:/Users/mishrd5/Downloads/demo");

**try** {

System.***out***.println("File Name: " + file.getOriginalFilename());

**file.transferTo(new File(fileStorageLocation + "/" + file.getOriginalFilename()));**

} **catch**(Exception ex) {

ex.printStackTrace();

}

**return** ResponseEntity.*ok*("File uploaded successfully...");

}

@PostMapping("/upload/file1")

**public** ResponseEntity<String> uploadFile11(@RequestParam("file") MultipartFile file) {

Path fileStorageLocation = Paths.*get*("C:/Users/mishrd5/Downloads/demo");

**try** {

System.***out***.println("File Name: " + file.getOriginalFilename());

**byte**[] bytes = file.getBytes();

**Path path = Paths.*get*(fileStorageLocation + "/" + file.getOriginalFilename());**

**Files.*write*(path, bytes);**

} **catch**(Exception ex) {

ex.printStackTrace();

}

**return** ResponseEntity.*ok*("File uploaded successfully...");

}

@PostMapping("/upload/image")

**public** ResponseEntity<String> uploadImage(@RequestPart("file") MultipartFile file) {

Path fileStorageLocation = Paths.*get*("C:/Users/mishrd5/Downloads/demo");

**try** {

System.***out***.println("File Name: " + file.getOriginalFilename());

file.transferTo(**new** File(fileStorageLocation + "/" + file.getOriginalFilename()));

} **catch**(Exception ex) {

ex.printStackTrace();

}

**return** ResponseEntity.*ok*("Image File uploaded successfully...");

}

Note:

* **Input Parameter can be both below.**

**@RequestParam("file") MultipartFile file**

**or**

**@RequestPart("file") MultipartFile file**

* **To store the file or to write the content, you can use both**

**String filePathToWrite = fileLocation + "/" + file.getOriginalFilename();**

**file.transferTo(new File(filePathToWrite));**

**or**

**byte[] bytes = file.getBytes(); 🡸 Input MultipartFile**

**Path path = Paths.get(fileLocation + "/" + file.getOriginalFilename());**

**Files.write(path, bytes);**

**How to upload a file along with contents**

@PostMapping("/upload/profile")

**public** ResponseEntity<String> uploadProfile(

@RequestPart("file") MultipartFile file,

@RequestPart String coverLetter) {

System.***out***.println("Cover Letter: " + coverLetter);

Path fileStorageLocation = Paths.*get*("C:/Users/mishrd5/Downloads/demo");

**try** {

System.***out***.println("File Name: " + file.getOriginalFilename());

String filePathToWrite = fileStorageLocation + "/" + file.getOriginalFilename();

**byte**[] bytes = file.getBytes();

Path path = Paths.*get*(filePathToWrite);

Files.*write*(path, bytes);

} **catch**(Exception ex) {

ex.printStackTrace();

}

**return** ResponseEntity.*ok*("Profile File uploaded successfully...");

}

Notes:

**@RequestPart("file") MultipartFile file**

**@RequestPart String coverLetter**