

WAYS TO WORK WITH RESPONSE ENTITY

```
return new ResponseEntity<>(Student, HttpStatus.CREATED);

return ResponseEntity.status(HttpStatus.CREATED).body(Student);

return ResponseEntity.ok(Student);

return ResponseEntity.of(Optional.of(Student));

return new ResponseEntity<Student>(HttpStatus.CREATED);
```

METHOD NAME	STATUS CODE
.ACCEPTED	202
.CREATED	201
.NOCONTENT	204
.OK	200

1. Basic ResponseEntity with Body

```
@GetMapping("/hello")

Public ResponseEntity<String> hello(){
    return ResponseEntity.ok("Hello world");
}
```

2. Response Entity with Custom HTTP Status

```
@GetMapping("/not found")

Public ResponseEntity<String> notFound(){

return new ResponseEntity<>("Resource not found", HttpStatus.NOT_FOUND);

}
```

3. Response Entity with Custom Object DTO

```
@GetMapping("/user/{id}")

Public ResponseEntity<User> getUser(@PathVariable Long id){

User user = userService.getUserById(id);

    return ResponseEntity.ok(user);

}
```

4. Response Entity for POST with CREATED(201)

```
@PostMapping("/user/create")

Public ResponseEntity<User> createUser(@RequestBody User user){

User user = userService.createUser(user);

    return new ResponseEntity<>(user, HttpStatus.CREATED);

}
```

5. Response Entity with DELETE(204)

```
@DeleteMapping("/user/delete/{id}")

Public ResponseEntity<Void> createUser(@PathVariable Long id){

User user = userService.deleteUserById(id);

    return new ResponseEntity.noContent().build();    //noContent represents 204

}
```

6. Response Entity with PUT(200)

```
@PutMapping("/user/update/{id}")

Public ResponseEntity<User> createUser(@PathVariable Long id, @RequestBody updatedUser){

User user = userService.updateUser(id,updatedUser);

    return new ResponseEntity.status(HttpStatusCode.valueOf(200)).body(user);

}
```

BASIC PRODUCT CRUD RESPONSE ENTITY EXAMPLE

Operation	Example(usage)
POST [Create a Product]	@PostMapping("/create") public ResponseEntity<Product> saveProduct(@RequestBody Product product) { Product savedProduct = productService.saveProduct(product); return ResponseEntity.status(HttpStatus.CREATED).body(savedProduct); }

GET [Get all products]	<pre>@GetMapping("/all") public ResponseEntity<List<Product>> getAllProducts() { List<Product> allProducts = productService.getAllProduct(); return ResponseEntity.status(HttpStatus.OK).body(allProducts); }</pre>
GET [Get product by id]	<pre>@GetMapping("/{id}") public ResponseEntity<Product> getProductById(@PathVariable Long id) { Product found = productService.getProductById(id); return ResponseEntity.status(HttpStatus.OK).body(found); }</pre>
PUT [Update a product by id]	<pre>@PutMapping("/{id}") public ResponseEntity<Product> updateProductById(@PathVariable Long id, Product updatedProduct) { return ResponseEntity.status(HttpStatus.OK).body(productService.updateProductById(id, updatedProduct)); }</pre> <p>OR WITH CUSTOM STATUS CODE</p> <pre>@PutMapping("/{id}") public ResponseEntity<Product> updateProductById(@PathVariable Long id, Product updatedProduct) { return ResponseEntity.status(HttpStatus.SC_200).body(productService.updateProductById(id, updatedProduct)); }</pre>
DELETE [Delete a product by id]	<pre>@DeleteMapping("/{id}") public ResponseEntity<Product> deleteProductById(@PathVariable Long id) { productService.deleteProductById(id); return ResponseEntity.status(HttpStatus.OK).body(null); }</pre> <p>OR WITH CUSTOM STATUS CODE</p> <pre>@DeleteMapping("/{id}") public ResponseEntity<Void> deleteProductById(@PathVariable Long id) { productService.deleteProductById(id); return ResponseEntity.status(HttpStatus.SC_204).body(null); }</pre> <p>RECOMMENDED WAY</p> <pre>@DeleteMapping("/{id}") public ResponseEntity<HttpStatus> deleteProductById(@PathVariable Long id) { productService.deleteProductById(id); return ResponseEntity.<>(HttpStatus.NO_CONTENT); }</pre>

USAGE OF RESPONSE ENTITY WITH DTO

With DTO + Service Layer (Recommended)

User Controller	UserDTO Class	User Service
<pre>@RestController @RequestMapping("/users") Public class UserController{ @Autowired Private UserService service; @GetMapping("/{id}") Public ResponseEntity<UserDTO> getUser(@PathVariable Long id){ UserDTO userDTO = service.getUserById(id); return ResponseEntity.ok(userDTO); } }</pre>	<pre>Public class UserDTO{ Private Long id; Private String name; Public UserDTO(User user){ this.id = user.getId(); this.name = user.getName(); this.email = user.getEmail(); } }</pre>	<pre>@Service Public class UserService{ @Autowired Private UserRepository userRepo; Public UserDTO getUserById(Long id){ User user = userRepo.findById(id).orElseThrow(()-> new RuntimeException("User not found")); return new UserDTO(user); } }</pre>

Without DTO + Service Layer

User Controller	User Service
<pre>@RestController @RequestMapping("/users") Public class UserController{ @Autowired Private UserService service; @GetMapping("/{id}") Public ResponseEntity<User> getUser(@PathVariable Long id){ User user = service.getUserById(id); return ResponseEntity.ok(user); } }</pre>	<pre>@Service Public class UserService{ @Autowired Private UserRepository userRepo; Public User getUserById(Long id){ User user = userRepo.findById(id).orElseThrow(()-> new RuntimeException("User not found")); return new User(user); } }</pre>