ResponseEntity

Certified Developer
The Ultimate Tech Degree

DigitalHouse >

ResponseEntity

Estende ou herda da classe HttpEntity e adiciona o HttpStatus, código de status. Normalmente é usado em serviços REST dentro dos métodos do controller.

O ResponseEntity lida com toda a respostas HTTP, incluindo o **body** (corpo), o **header** (cabeçalho) e o **código de status**, permitindo total liberdade para configurar a resposta que queremos enviar para nossos endpoints.

Devido ao fato do tipo de parâmetro ser genérico, podemos fazer o seguinte:

```
@GetMapping(“/ola”)
ResponseEntity<String> olaMundo() {
     return new ResponseEntity <>(“Olá mundo de uma resposta HTTP!”,
                                        HttpStatus.OK) ;
}
```

Conseguimos retornar uma String como body (corpo), e também devolvemos junto com a resposta o código de status 200 OK.

Agora podemos programar nosso endpoint para retornar um código de status 400, solicitação incorreta, no caso de inserir um e-mail inválido:

```
@GetMapping(“/verificar/{email}”)
ResponseEntity<String> verificarEmail(@PathVariable String email) {

     if (!EmailValidator.getInstance().isValid(email) ) {
          return new ResponseEntity <>(“O formato deve ser:
                                        exemplo@dominio.com”,
                                        HttpStatus.BAD_REQUEST) ;
     }

     return new ResponseEntity<> ( “Seu e-mail é: ” + email, HttpStatus.OK) ;
}
```

Além disso, podemos definir o header (cabeçalho) de resposta da seguinte maneira:

```
@GetMapping(“/cabecalho/{cliente}”)
ResponseEntity<String> cabecalhoPersonalizado(@PathVariable String cliente) {

     HttpHeaders cabecalho = new HttpHeaders ();

     cabecalho.add(“Status do cliente”,
                   “Cliente ” + cliente + “: habilitado”);

     return new ResponseEntity <>(“Bem-vindo ” + cliente, cabecalho,
                                  HttpStatus.OK);
}
```

Por fim, o ResponseEntity fornece duas classes do tipo interface: BodyBuilder e HeadersBuilder. O ResponseEntity tem um método estático para acessar essas interfaces.

Veja seu uso em nosso exemplo Olá Mundo:

```
@GetMapping("/OlaMundo2")
ResponseEntity<String> OlaMundo2() {

        return ResponseEntity.ok("Olá Mundo!");
}
```

Para o caso da validação do e-mail:

```
@GetMapping(“/verificar2/{email}”)
ResponseEntity<String> verificarEmail2(@PathVariable String email) {

     if (!EmailValidator.getInstance().isValid(email) ) {
          return ResponseEntity.badRequest().body(“O formato de ser:
                                                  exemplo@dominio.com”);
     }

     return ResponseEntity.status(200).body(“Seu e-mail é: ” + email);
}
```

Por último, personalizar o cabeçalho:

```
@GetMapping(‘’/cabecalho2/{cliente}’’)
ResponseEntity<String> cabecalhoPersonalizado2(@PathVariable String cliente) {

        return ResponseEntity.ok()
                             .header(“Status do cliente”,
                                     “Cliente ” + cliente + “: habilitado”)
                             .body(“Bem-vindo ” + cliente);
}
```

DigitalHouse>