ResponseEntity

DigitalHouse>

Certified Tech Developer
The Ultimate Degree

ResponseEntity

Extiende o hereda de la clase HttpEntity y agrega un HttpStatus código de estado. Normalmente es usada en los servicios REST dentro de los métodos de los controladores.

ResponseEntity maneja toda la respuesta HTTP incluyendo el **cuerpo, cabecera y códigos de estado** permitiéndonos total libertad para configurar la respuesta que queremos que se envíe desde nuestros endpoints.

Gracias a que el tipo de parámetro es genérico podemos realizar lo siguiente:

```
@GetMapping(‘’/hola’’)
ResponseEntity<String> holaMundo() {
     return new ResponseEntity <>(‘’Hola Mundo desde una respuesta HTTP!’’,
                                        HttpStatus.OK) ;
}
```

Logrando retornar como tipo de cuerpo un String, además entregamos junto a la respuesta un código de estado 200 OK.

Ahora podemos programar nuestro endpoint para que retorne un código 400 de petición mala en caso de ingresar un correo inválido:

```
@GetMapping(‘’/verificar/{correo}’’)
ResponseEntity<String> verificarCorreo(@PathVariable String correo) {

     if (!EmailValidator.getInstance().isValid(correo) ) {
          return new ResponseEntity <>(‘’Formato debe ser:
                                        ejemplo@correo.dominio’’,
                                        HttpStatus.BAD_REQUEST) ;
     }

     return new ResponseEntity<> ( ‘’Su correo es: ‘’+ correo, HttpStatus.OK) ;
}
```

Además, podemos establecer la cabecera de la respuesta de la siguiente manera:

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

     HttpHeaders cabecera = new HttpHeaders ();

     cabecera.add( ‘’Estado Cliente’’,
                   ‘’Cliente’’+ cliente + ‘’: habilitado’’ );

     return new ResponseEntity <>(‘’Bienvenido ’’ + correo, cabecera,
                                       HttpStatus.OK);
}
```

Finalmente, ResponseEntity provee dos clases anidadas de tipo interface: BodyBuilder y HeadersBuilder. ResponseEntity posee un método estático para poder acceder a estas interfaces.

Para nuestro ejemplo Hola Mundo:

```
@GetMapping(‘’/HolaMundo’’)
ResponseEntity<String> HolaMundo2() {

        return ResponseEntity.ok(‘’HolaMundo !’’);
}
```

Para el caso de validación de correo:

```
@GetMapping(''/verificarCorreo/{correo}'')
ResponseEntity<String> verificarCorreo2(@PathVariable String correo) {

     if (!EmailValidator.getInstance().isValid(correo) ) {
          return ResponseEntity.badRequest().body(''Error ! Formato correcto:
                                                  ejemplo@correo.dominio'');
     }

     return ResponseEntity.status(200).body(''Correo: ''+ correo);
}
```

Por último, para realizar una cabecera personalizada:

```
@GetMapping(‘’/cabeceraCustomizada/{cliente}’’)
ResponseEntity<String> cabeceraPersonalizada2(@PathVariable String cliente) {

        return ResponseEntity.ok()
                            .header(‘’Estado Cliente’’,
                                    ‘’Cliente’’ + cliente ‘’: habilitado’’)

                            .body(Bienvenido cliente: ‘’ + cliente);
}
```

DigitalHouse>