



**DGTIC UNAM**  
DIRECCIÓN GENERAL DE CÓMPUTO Y  
DE TECNOLOGÍAS DE INFORMACIÓN  
Y COMUNICACIÓN



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO  
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS  
DE INFORMACIÓN Y COMUNICACIÓN



DIPLOMADO

# Desarrollo de sistemas con tecnología Java

## Módulo 10

API RESTful con Spring Boot

*M. en C. Jesús Hernández Cabrera*



# Personalizando respuestas REST

- **Código de estado.**- Es un número de 3 dígitos para indicar el resultado de un recurso solicitado. [Link a documentación.](#)
- **Encabezados** (headers).- Proporciona información adicional de la respuesta HTTP. Por ejemplo, Content-Type: application/json
- **Cuerpo.**- El contenido de la respuesta en el formato adecuado. Texto, Json, HTML, etc.

# Cabeceras HTTP importantes para API REST

Cabecera	Descripción	Ejemplo	Uso Principal
Accept	Indica el tipo de respuesta que el cliente espera recibir.	Accept: application/json	Filtra el tipo de respuesta (JSON, XML).
Content-Type	Indica el tipo de contenido presente en el cuerpo de la solicitud o respuesta.	Content-Type: application/json	Define el formato de los datos enviados.
Authorization	Contiene las credenciales para autenticar al cliente o autorización del usuario.	Authorization: Bearer <token>	Usado para <b>tokens</b> de API y JWT.
Cache-Control	Controla la forma en que la respuesta puede ser almacenada en cache por el cliente o intermediarios.	Cache-Control: no-cache	Gestiona el almacenamiento en caché.
User-Agent	Identifica la aplicación cliente que realiza la solicitud.	User-Agent: PostmanRuntime/7.29	Informar sobre el cliente HTTP.
Location	Indica la URL de un recurso recién creado o redireccionado.	Location: /api/resource/1	Usado en respuestas <b>201 Created</b> .
Content-Length	Indica la longitud del cuerpo en bytes.	Content-Length: 348	Permite saber cuánto se debe leer.
ETag	Identificador único de la versión del recurso para control de concurrencia.	ETag: "abc123"	Usado en <b>control de versiones</b> .
Accept-Encoding	Indica los algoritmos de compresión aceptados por el cliente.	Accept-Encoding: gzip, deflate	Optimiza transferencias con compresión.
Set-Cookie	Establece cookies en el cliente para sesiones u otras configuraciones.	Set-Cookie: sessionId=abc123	Manejo de <b>sesiones</b> en clientes.

# @GetMapping() y @ResponseStatus

- **value** o **path**: Establece la ruta del end-point.
- **params**: Establece que parámetros debe considerar.
- **headers**: **Restringe** el acceso a la respuesta solo a los encabezados establecidos. Se espera que el **cliente** Web envíe ese encabezado.
- **produces**: Define el tipo de contenido (MIME Type) que el endpoint generará como respuesta.
- **consumes**: Define el tipo de contenido del cuerpo de la solicitud que el servidor puede aceptar.

# Ejemplo de path, produces y headers

```
public LibroRestController() {  
    libreria = new HashMap<>();  
    libreria.put(1, new Libro(1, "El perfume", "Patrik Süskind"));  
    libreria.put(2, new Libro(2, "El señor de los anillos", "J. R. Tolkien"));  
    libreria.put(3, new Libro(3, "Fundación", "Isaac Asimov"));  
}  
  
//@GetMapping("/")  
@GetMapping(path = "/", headers = {"Accept=application/json"},  
             produces = MediaType.APPLICATION_JSON_VALUE)  
public HashMap<Integer, Libro> getAll() {  
    return libreria;  
}
```

Preparativos / <http://127.0.0.1:8080/libreria/>


GET ▼ <http://127.0.0.1:8080/api/libreria/>

Params Authorization **Headers (9)** Body Scripts Settings

<input type="checkbox"/>	Accept	*/*
<input checked="" type="checkbox"/>	Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/>	Connection	keep-alive
<input type="checkbox"/>	Content-Type	text/*
<input checked="" type="checkbox"/>	Accept	application/json

Body Cookies **Headers (5)** Test Results

200 OK • 12 m

Pretty Raw Preview Visualize JSON ▼ 

```
1 {  
2   "1": {  
3     "id": 1,  
4     "titulo": "El perfume",  
5     "autor": "Patrik Süskind"  
6   },
```

# Hacer el resto de los métodos

```
@GetMapping(value =("/{id}", produces = MediaType.APPLICATION_JSON_VALUE)  
public Libro getLibro(@PathVariable int id) {  
    return libreria.get(id);  
}
```

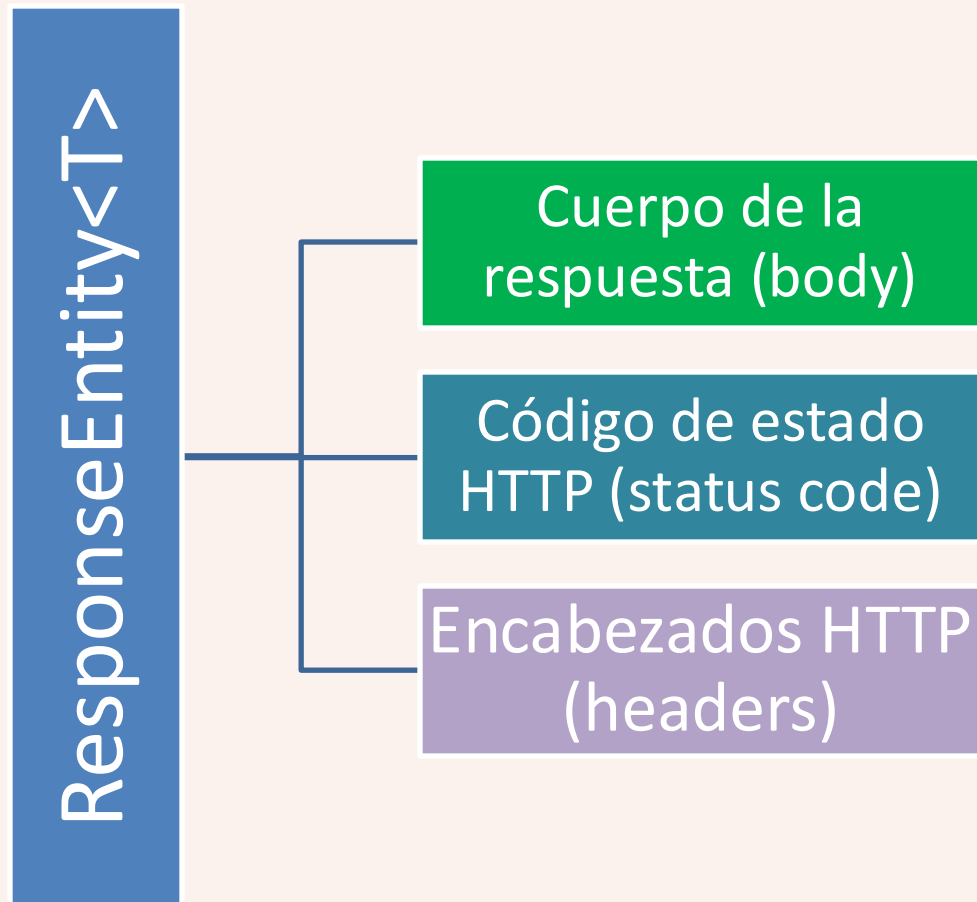


# Personalizando el código de estado. HTTP Status code

- Por defecto los verbos @GetMapping, @PostMapping, etc. Retornan:
  - Un código 200 (OK) si todo salió bien
  - Un código 500 si existe una excepción del lado del servidor.
- Podemos cambiar el código de estado y personalizar la respuesta con la clase **ResponseEntity<T>**.



# ResponseEntity<T>



- Mayor control sobre las respuestas que enviamos desde un controlador REST.
- Flexibilidad para personalizar el resultado.

```
ResponseEntity<T> responseEntity = new ResponseEntity<>(body, headers, status);
```

# Ejemplo práctico.

```
23
24 // @GetMapping("/")
25 @GetMapping(path = "/", headers = {"Accept=application/json"},
26             produces = {MediaType.APPLICATION_JSON_VALUE,
27                          MediaType.APPLICATION_XML_VALUE
28                          })
29 public ResponseEntity<HashMap<Integer, Libro>> getAll() {
30     //return libreria;
31
32     return new ResponseEntity<>(libreria, HttpStatus.OK);
33 }
```

# Métodos Estáticos de ResponseEntity

- Son atajos que proporciona la clase ResponseEntity para crear instancias de manera más sencilla y directa.
- Facilitan la construcción de respuestas HTTP con los códigos de estado más comunes sin tener que crear manualmente un ResponseEntity usando el constructor.

```
return ResponseEntity.ok(libro); // 200 OK con un cuerpo  
return ResponseEntity.ok().build(); // 200 OK sin cuerpo
```

# Get y ResponseEntity

```
@GetMapping(value =("/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Libro> getLibro(@PathVariable int id) {
    //return libreria.get(id);
    Libro libro = libreria.get(id); //servicio
    if (libro != null) {
        return ResponseEntity.ok(libro); //200
    } else {
        return ResponseEntity.notFound().build(); // 404
    }
}
```

# POST y ResponseEntity

```
@PostMapping(value = "/", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Libro> addBook(@RequestBody Libro libro) {
    int id = 1;
    while (libreria.containsKey(id)) { // Simular el autoincrement de BD
        id++;
    }
    libro.setId(id);
    libreria.put(libro.getId(), libro);
    return new ResponseEntity<>(libro, HttpStatus.CREATED);
}
```

# Put y ResponseEntity

```
// Reemplazar un recurso
@PutMapping(value=("/{id}" , produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Libro> reemplazarLibro(@PathVariable int id, @RequestBody Libro libro) {
    //libreria.replace(id, libro);
    //return libro;
    if (libreria.containsKey(id)) {
        libro.setId(id);
        libreria.replace(id, libro); // Sql Update
        //return ResponseEntity.noContent().build(); // 204
        return new ResponseEntity<>(libro, HttpStatus.OK);
    } else {
        int idNuevo = 1;
        while (libreria.containsKey(idNuevo)) { // Simular el autoincrement de BD
            idNuevo++;
        }
        libro.setId(idNuevo);
        libreria.put(libro.getId(), libro);
        return new ResponseEntity<>(libro, HttpStatus.CREATED);
    }
}
```

# Patch y ResponseEntity

```
85
86     @PatchMapping(value =("/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
87     public ResponseEntity<Libro> actualizarLibro(@PathVariable int id, @RequestBody Libro libro) {
88         Libro dbLibro = libreria.get(id); // Simular un SQL Update
89         if (dbLibro==null){
90             return ResponseEntity.notFound().build();
91         }
92
93         if (libro.getAutor() != null) {
94             dbLibro.setAutor(libro.getAutor());
95         }
96         if (libro.getTitulo() != null) {
97             dbLibro.setTitulo(libro.getTitulo());
98         }
99
100        libreria.replace(id, dbLibro);
101        return ResponseEntity.ok(dbLibro);
102    }
```



# Delete

```
@DeleteMapping(value =("/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Libro> eliminaLibro(@PathVariable int id) {
    //return libreria.remove(id);
    if (libreria.containsKey(id)) {
        libreria.remove(id);
        return ResponseEntity.noContent().build(); //204
    } else {
        return ResponseEntity.notFound().build(); //404
    }
}
```

# Participación: grupos pequeños

- Trabajar en pequeños grupos en Zoom para replicar el proyecto.
- Leer las instrucciones del documento “Participación 1.pdf” disponible en la sección actividades.

# Contacto

*M. En C. Jesús Hernández Cabrera*  
*Profesor de carrera*

jesushc@unam.mx

Redes sociales:



[www.linkedin.com/in/hcjesus](https://www.linkedin.com/in/hcjesus)