

DIPLOMADO

Desarrollo de sistemas con tecnología Java

Módulo 10

API RESTful con Spring Boot

M. en C. Jesús Hernández Cabrera



Anotaciones

@RestController

- Se utiliza en el nivel de clase para indicar que la clase es un controlador donde cada método devuelve un objeto de dominio en lugar de una vista.
- Combina @Controller y @ResponseBody.

@RequestMapping

- Se puede usar en el nivel de clase o de método para mapear solicitudes web a clases o métodos específicos del controlador.
- Puede especificar el URL, el método HTTP, los parámetros, los encabezados, y los tipos de medios consumidos o producidos.

Anotaciones

`@GetMapping,`
`@PostMapping,`
`@PutMapping,`
`@DeleteMapping,`
`@PatchMapping`

- Estas anotaciones son especializaciones de `@RequestMapping` que simplifican la configuración al especificar directamente el tipo de operación HTTP (GET, POST, PUT, DELETE, PATCH).
- Son convenientes para mapear los métodos del controlador a las operaciones RESTful.

Prefacio

- En el resto de las diapositivas exploraremos una implementación simple de los verbos REST.
- El objetivo es comprender el núcleo de REST con un ejemplo sencillo.
- Posteriormente, se abordarán temas satélite a los aquí presentados, como por ejemplo, códigos de estado, respuestas personalizadas y manejo de errores.

@GetMapping. (all)

- **GET**

- Se utiliza para recuperar representaciones de recursos.
- No debe tener efectos secundarios, lo que significa que no modifica el estado del recurso.
- Las consultas son ***idempotentes***; hacer una solicitud GET múltiples veces no cambiará el resultado.
- En Spring, se utiliza la anotación **@GetMapping**

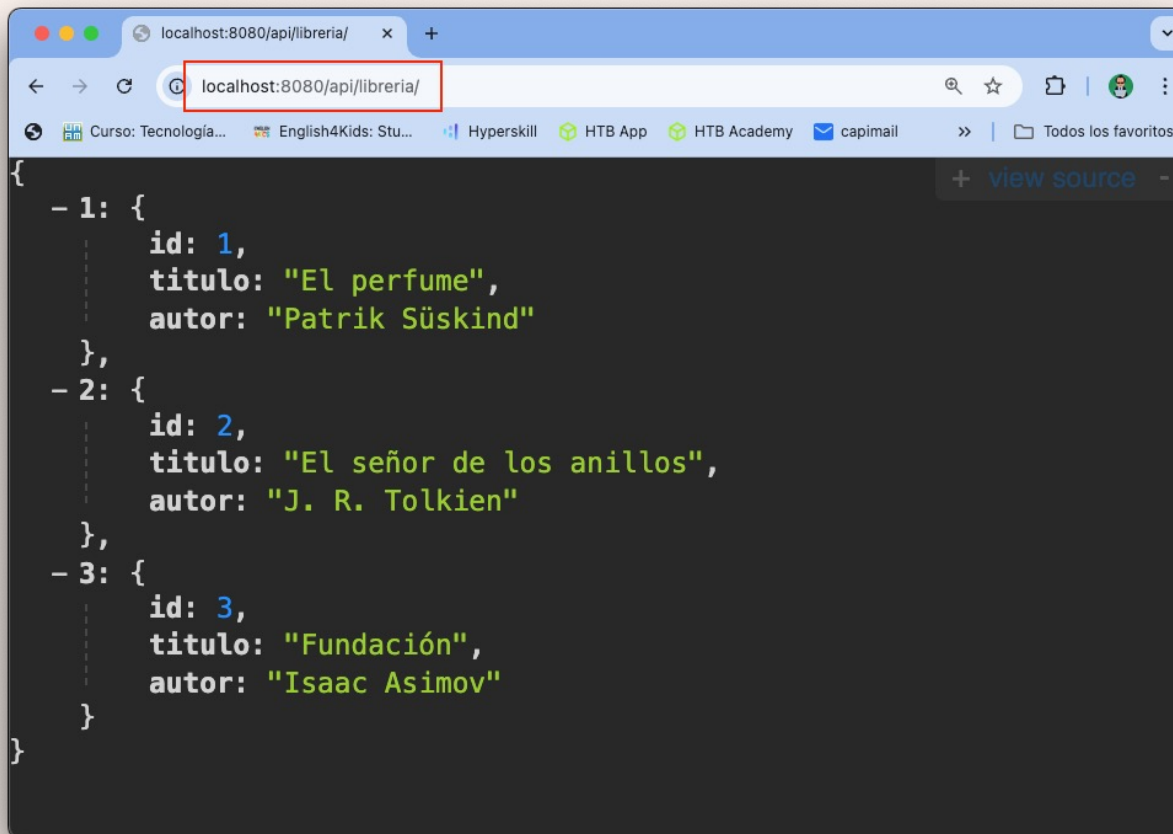
@GetMapping, Ejemplo de obtener todos los registros.

```
Libro.java x LibroRestController.java x
@RestController
@RequestMapping("/api/libreria")
public class LibroRestController {
    HashMap<Integer, Libro> libreria;

    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("/")
    public HashMap<Integer, Libro> getAll() {
        return libreria;
    }
}
```

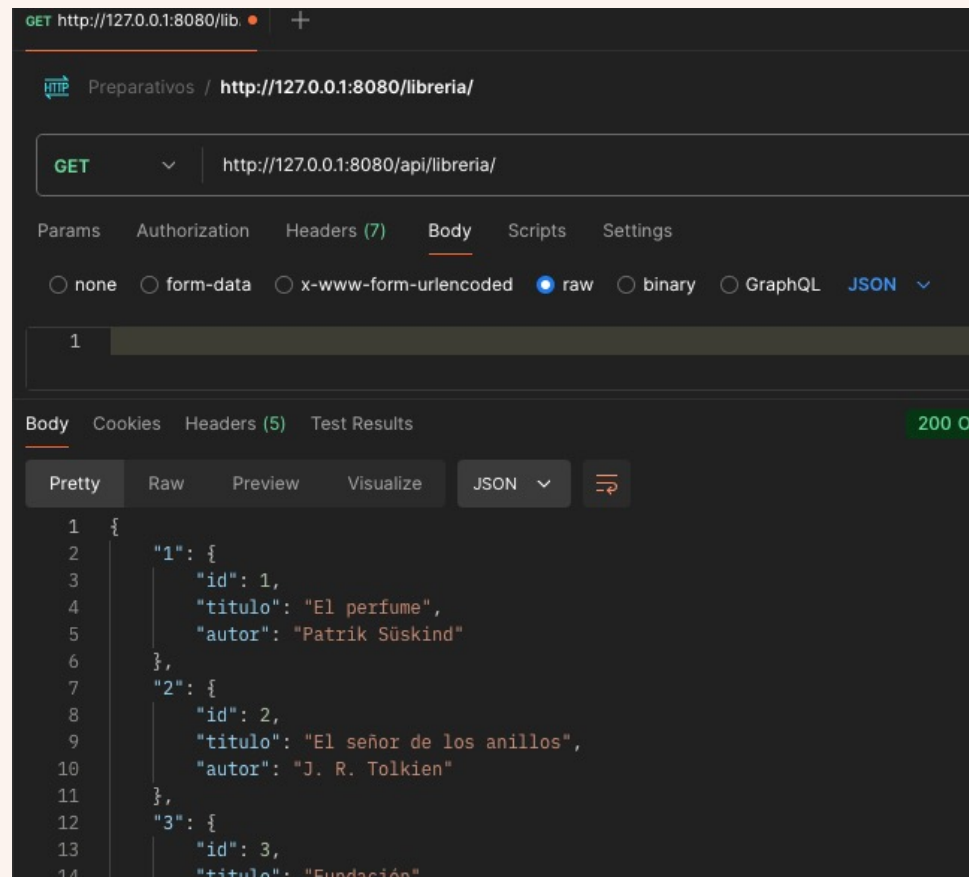
GetAll



A screenshot of a web browser window displaying a JSON response. The address bar shows the URL `localhost:8080/api/libreria/`, which is highlighted with a red rectangle. The browser's tab bar shows several tabs, including "Curso: Tecnología...", "English4Kids: Stu...", "Hyperskill", "HTB App", "HTB Academy", and "capimail". The main content area displays a JSON array of three book objects. The browser's developer tools are open, showing the JSON data in a dark-themed editor. A "view source" link is visible in the top right corner of the developer tools.

```
{
  - 1: {
    id: 1,
    titulo: "El perfume",
    autor: "Patrik Süskind"
  },
  - 2: {
    id: 2,
    titulo: "El señor de los anillos",
    autor: "J. R. Tolkien"
  },
  - 3: {
    id: 3,
    titulo: "Fundación",
    autor: "Isaac Asimov"
  }
}
```

Desde PostMan



Anotaciones

@PathVariable

- Se utiliza en el parámetro de un método del controlador para vincular una variable de la URL ({variable}) a un parámetro del método.
- Es útil para manejar valores dinámicos en las rutas de los endpoints.

@RequestParam

- Se utiliza en el parámetro de un método del controlador para vincular parámetros de la solicitud HTTP a un parámetro del método.
- Es útil para acceder a los datos enviados en la cadena de consulta de la URL.

@GetMapping (Obtener uno por Id)

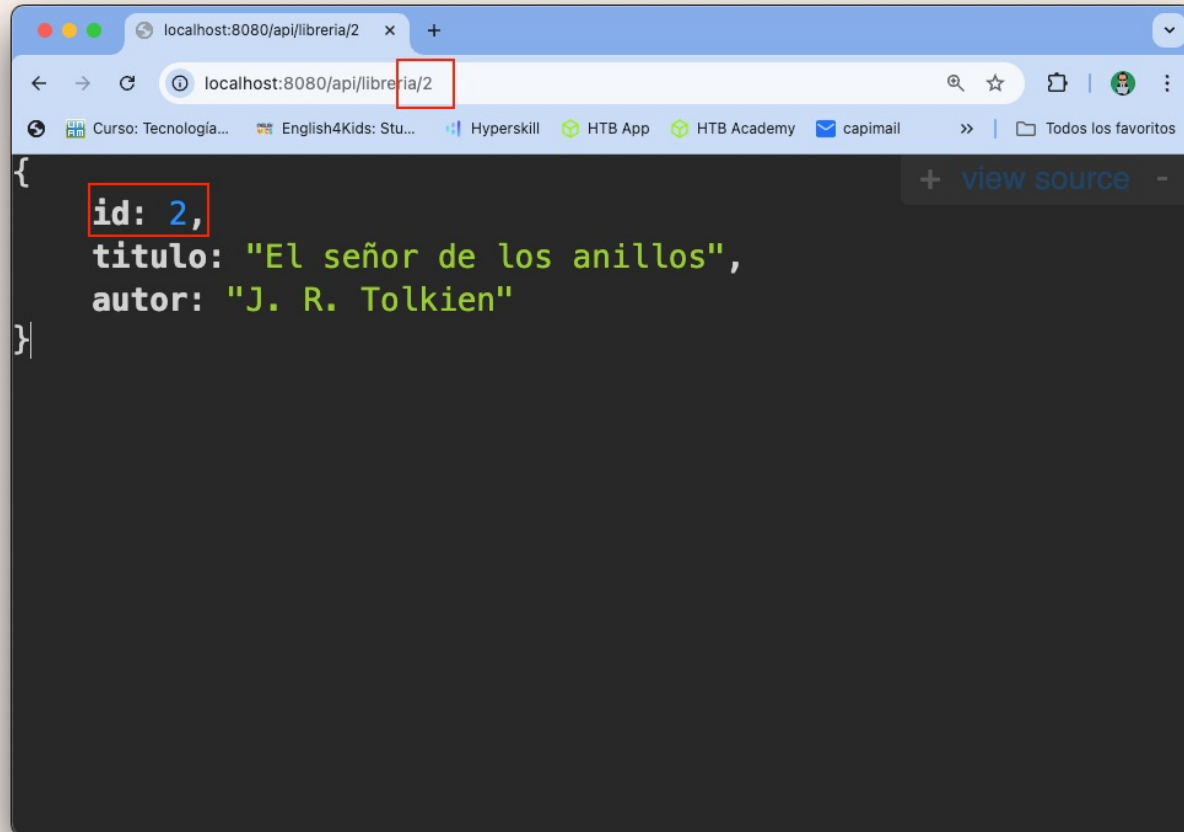
- **GET**

- Se utiliza para recuperar una tupla específica, seleccionada por un campo, comúnmente el **Id**.
- No debe tener efectos secundarios y es *idempotentes*.
- En Spring, se utiliza:
 - La anotación **@GetMapping("/api/librería/{id}")**.
 - La sintaxis **{id}** en la ruta del mapeo.
 - La anotación **@PathVariable** para recuperar el valor.

@GetMapping Ejemplo de obtener un registro.

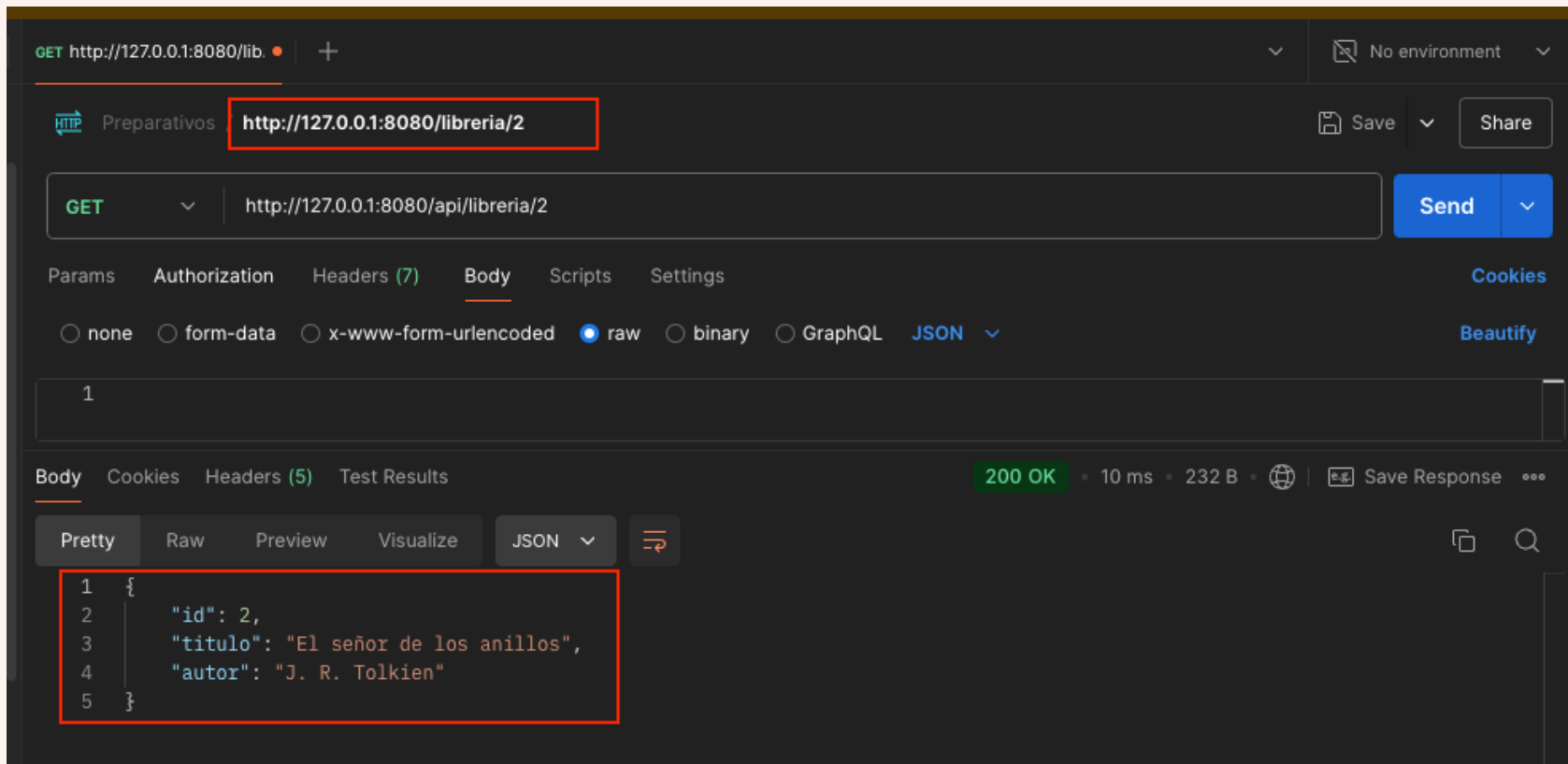
```
22
23     @GetMapping("/")
24     public HashMap<Integer, Libro> getAll() {
25         return libreria;
26     }
27
28
29     @GetMapping("/{id}")
30     public Libro getLibro(@PathVariable int id) {
31         return libreria.get(id);
32     }
33
```

Get One



```
{  
  id: 2,  
  titulo: "El señor de los anillos",  
  autor: "J. R. Tolkien"  
}
```

Get One (postMan)



Anotaciones

@RequestBody

- Se utiliza en el parámetro de un método del controlador para indicar que el cuerpo de la solicitud HTTP debe ser vinculado a ese parámetro.
- Spring convierte automáticamente el cuerpo de la solicitud a un objeto Java utilizando convertidores de mensajes HTTP.

@ResponseBody

- Se utiliza en el nivel del método para indicar que el resultado del método debe ser vinculado al cuerpo de la respuesta HTTP.
- Cuando se usa @RestController, esta anotación se aplica implícitamente a todos los métodos.

@PostMapping

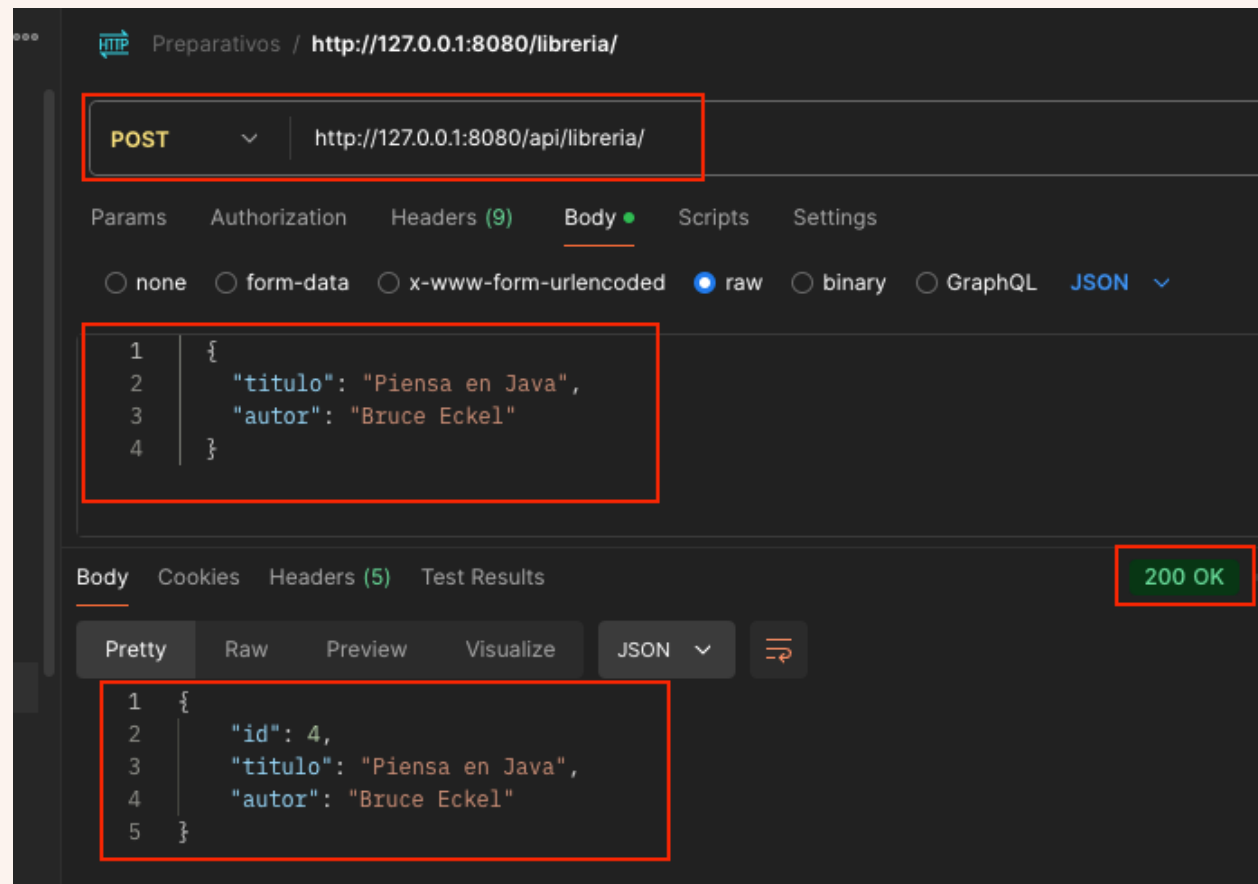
- **POST**

- Se emplea para crear nuevos recursos.
- Es común enviar datos en formatos JSON o XML en el cuerpo de la solicitud.
- En Spring, se usa la anotación **@PostMapping**
- **@RequestBody** se utiliza para mapear el cuerpo de la solicitud a un objeto Java.
- Los datos del lado del cliente pueden provenir de un formulario HTML o en formato text (RAW).

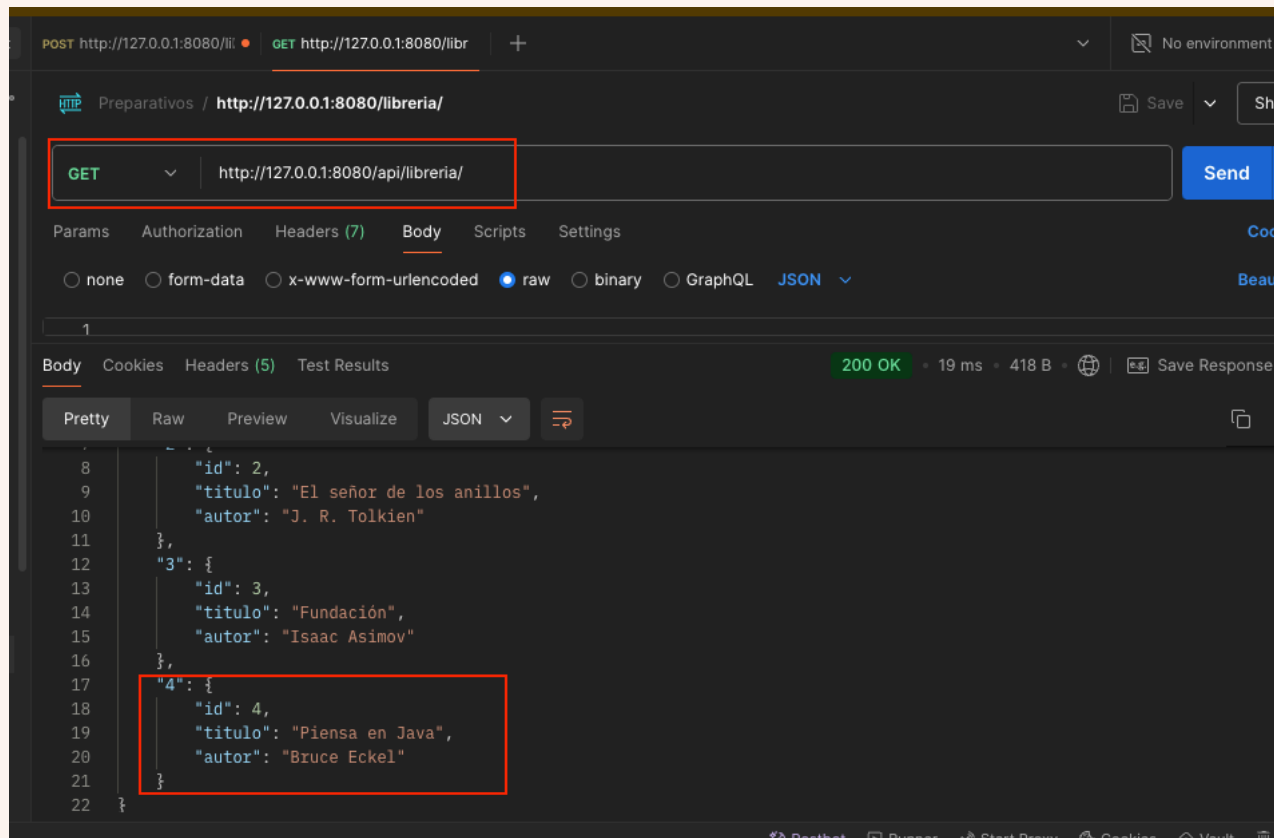
@PostMapping, Ejemplo para crear un nuevo registro.

```
}  
  
@PostMapping("/")  
public 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 libro;  
}
```


Post, crear un nuevo registro.



Comprobar con Get /



Métodos GET,POST,PUT,DELETE en HTTP

- PUT

- Se asocia con la actualización **completa** de recursos existentes.
- Al igual que POST, el cuerpo de la solicitud puede ser JSON o XML.
- **@PutMapping** es la anotación utilizada en Spring
- **@RequestBody** mapea el cuerpo de la solicitud al objeto de actualización.
- Requiere la identificación del registro con @PathVariable.
- Si el recurso no existe podría crearlo, dependiendo de la implementación.

@PutMapping. Reemplazar un recurso completo.

```
// Reemplazar un recurso
@PutMapping("/{id}")
public Libro reemplazarLibro(@PathVariable int id, @RequestBody Libro libro) {
    libreria.replace(id, libro);
    return libro;
}
```


Preparativos / <http://127.0.0.1:8080/libreria/2> Save Share


PUT ▼ <http://127.0.0.1:8080/api/libreria/2> Send ▼

Params Authorization Headers (9) **Body** • Scripts Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   "id": 2,
3   "titulo": "EL SEÑOR DE LOS ANILLOS",
4   "autor": "JR TOLKIEN"
5 }
6
```

Body Cookies Headers (5) Test Results 200 OK • 57 ms • 229 B •  Save Response ...

Pretty Raw Preview Visualize **JSON** ▼ 

```
1 {
2   "id": 2,
3   "titulo": "EL SEÑOR DE LOS ANILLOS",
4   "autor": "JR TOLKIEN"
5 }
```

@PatchMapping, Modificar parcialmente.

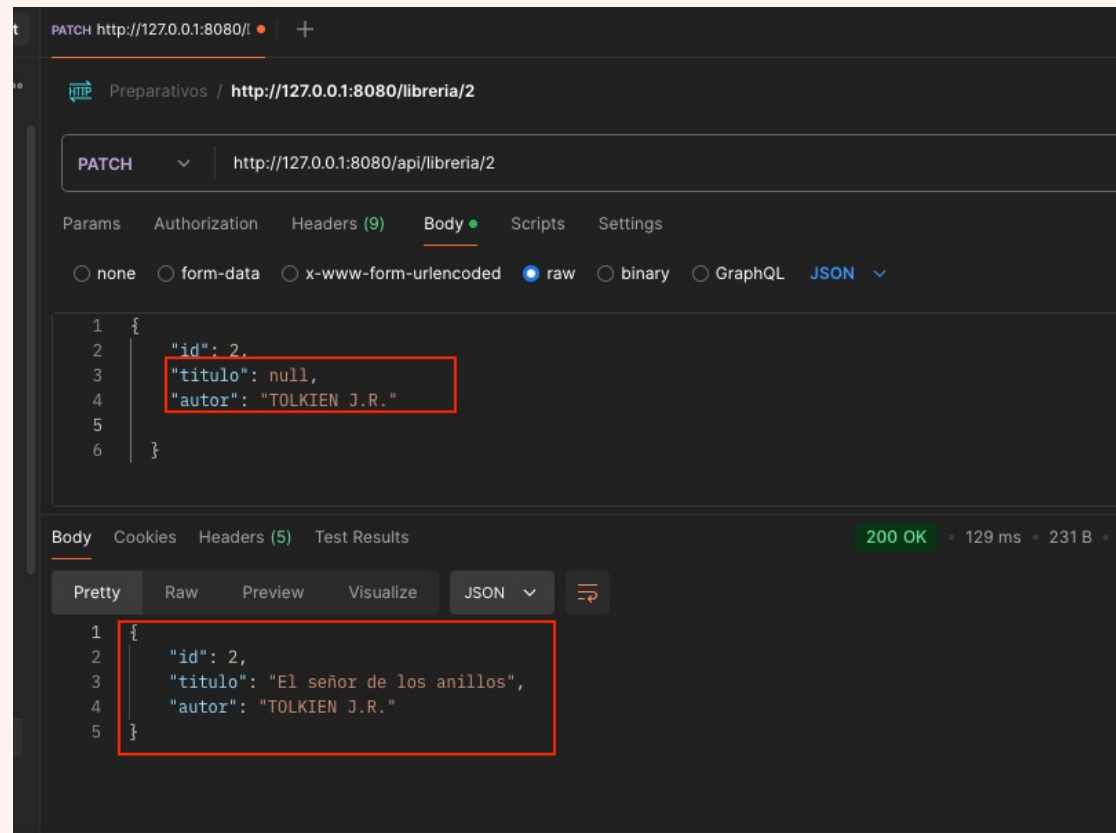
- PATCH
 - Modifica solo algunos de los campos del registro.
 - Al igual que POST, el cuerpo de la solicitud puede ser JSON o XML.
 - **@PatchMapping** es la anotación utilizada en Spring
 - **@RequestBody** mapea el cuerpo de la solicitud al objeto de actualización.
 - Requiere la identificación del registro con @PathVariable.

Patch, modificación parcial.

```
@PatchMapping("/{id}")
public Libro actualizarLibro(@PathVariable int id, @RequestBody Libro libro) {
    Libro dbLibro = libreria.get(id); // Simular un SQL Update
    if (libro.getAutor() != null) {
        dbLibro.setAutor(libro.getAutor());
    }
    if (libro.getTitulo() != null) {
        dbLibro.setTitulo(libro.getTitulo());
    }

    libreria.replace(id, dbLibro);
    return dbLibro;
}
```

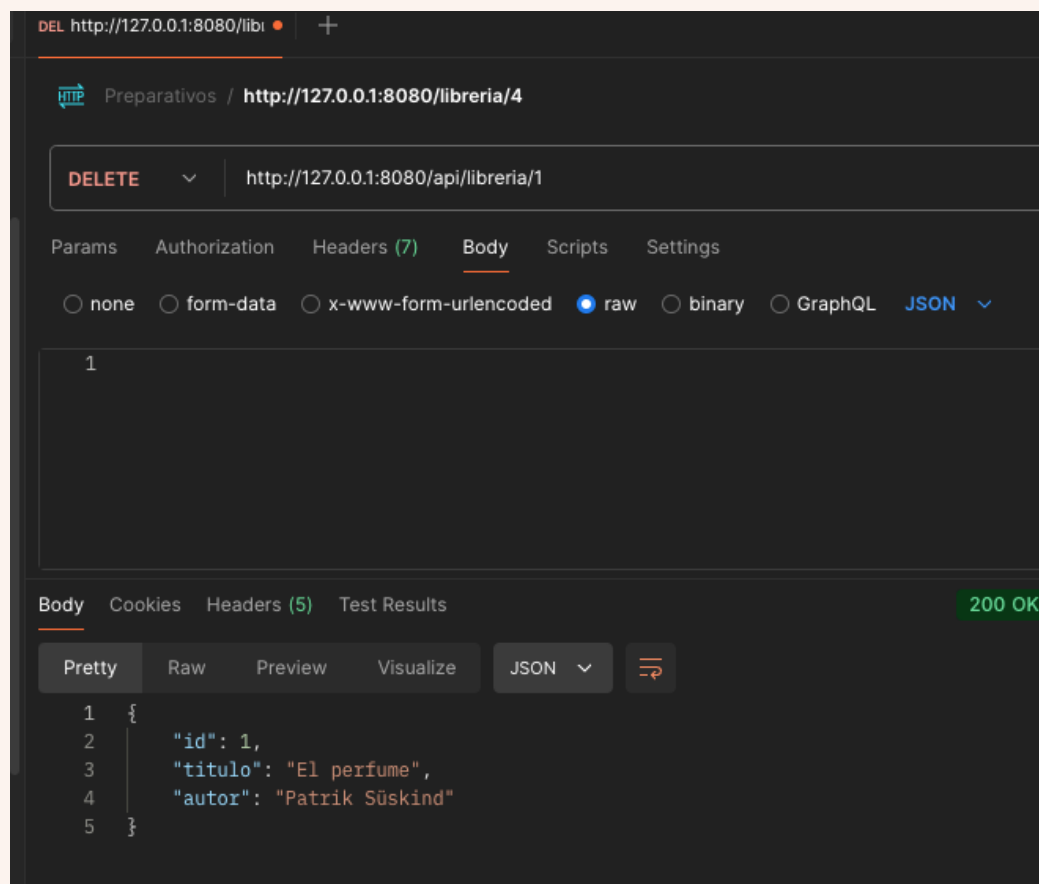
Actualización parcial (PostMan)



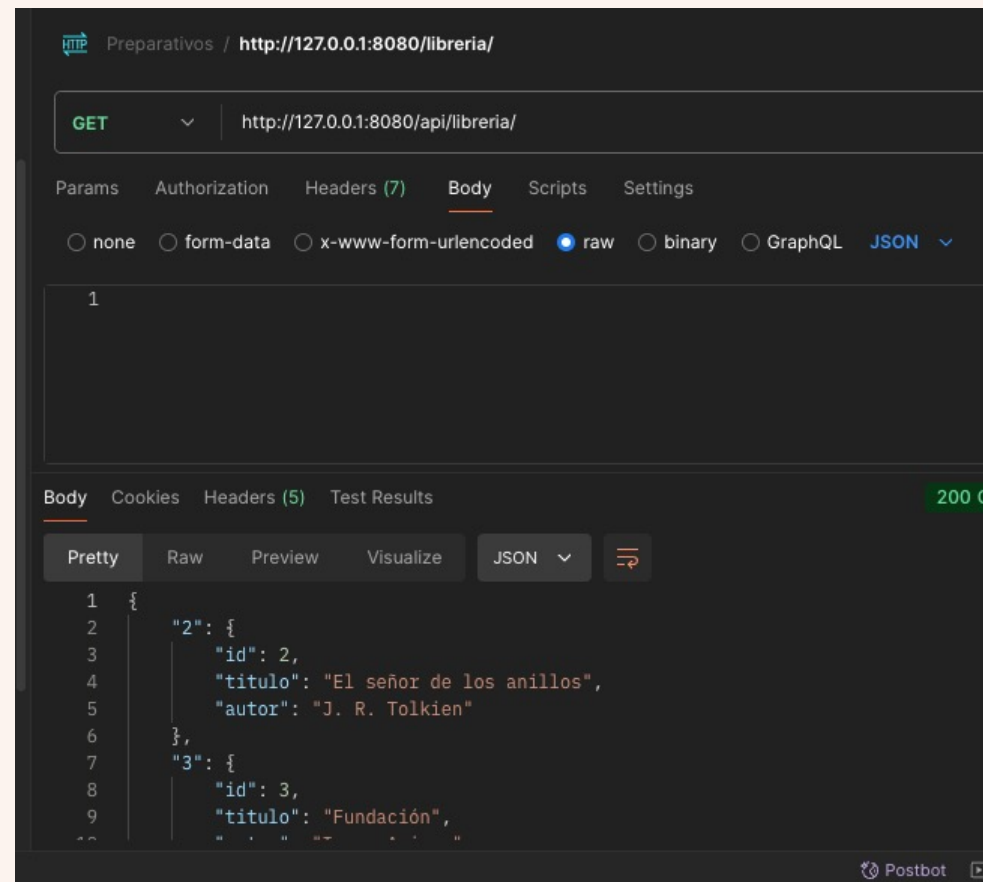
@DeleteMapping, Eliminar un recurso

- DELETE
 - Este método se utiliza para eliminar recursos.
 - Es una operación idempotente; eliminar el mismo recurso una y otra vez produce el mismo resultado que hacerlo una sola vez.
 - En Spring, se usa la anotación **@DeleteMapping**.
 - Usualmente, se pasa un identificador de recurso en la URL para especificar qué recurso eliminar, mapeando el identificador con **@PathVariable**

```
@DeleteMapping("/{id}")  
public Libro eliminaLibro(@PathVariable int id) {  
    return libreria.remove(id);  
}
```



Eliminar recurso (PostMan)



Resumen

Método HTTP	Operación CRUD	Respuestas HTTP Status, Elemento específico: (ejemplo: /users/{id})	Respuestas HTTP Status a la colección entera. (ejemplo: /users/)
POST	Create / insert	404	201 Created Res.: el documento con nuevo _id
GET	Read /Select	200 OK Resp.: Un sólo documento	/users/: 200 OK, Resp.: Todos los documentos
PUT	Replace / Update	200 OK ó 204 (Body sin contenido). Ó 404 (id no encontrado)	405 acción no permitida.
PATCH	Update Only	200 (OK) or 204 (Body sin contenido). 404 (No encontrado)	405 acción no permitida.
DELETE	Delete	200 (OK) ó 404 (No encontrado)	405 acción no permitida.

Contacto

M. en C. Jesús Hernández Cabrera
Profesor de carrera

jesushc@unam.mx

Redes sociales:



www.linkedin.com/in/hcjesus