

Anotaciones en el controller

DigitalHouse>



**Certified Tech
Developer**

The Ultimate Degree

Índice

1. Múltiples parámetros con el método GET
2. Múltiples parámetros con el método POST

1 | Múltiples parámetros con el método GET

@GetMapping

El controlador es el artefacto central de nuestro servicio REST.

En un método dentro del controlador podemos recibir tantos parámetros como lo sea necesario.

Tomando como ejemplo un *HelloWorld*, si se desea obtener por parámetro además del nombre y también el apellido y la edad, haríamos lo siguiente:

```
@RestController
public class HelloRestController {
    @GetMapping(path = "{name}/{lastname}/{age}")
    public String sayHello( @PathVariable String name,
                           @PathVariable String lastName,
                           @PathVariable Integer age) {

        return "Hola, " + name + " " + lastName + " Tu edad es: " + age;
    }
}
```

@PathVariable

Es una anotación que permite extraer información que es parte de la estructura de la URI pero que no se trata como un par nombre=valor.

Tal como lo dice su nombre, es “variable”, por lo que el valor que sea ingresado en la URI será el que se asigne a la variable mapeada con esta anotación.

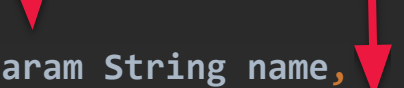
```
@GetMapping("/user/{userId}")  
public User getUser(@PathVariable("userId") String userId){  
    //...  
}
```

@RequestParam

Es una anotación que permite recibir parámetros desde una ruta mediante el método GET, para trabajar con ellos e incluso poder emitir una respuesta que dependa de los parámetros que sean obtenidos.

Cada uno de los parámetros generalmente se ubican en la URL después de un signo de pregunta "?" y están anidados por un "&".

Por ejemplo: <http://localhost:8080/student?name=Horacio&lastname=Quiroga>



```
@GetMapping(path = "/student/")  
public Student findStudent( @RequestParam String name,  
                             @RequestParam String lastName) {  
    //...  
}
```

@PathVariable vs @RequestParam

| @PathVariable | @RequestParam |
|--|--|
| Se usa para recuperar valores de la propia URI. | Se usa para recuperar parámetros de consulta. Los obtiene identificándolos luego del '?' en la URL. |
| Forma de la petición: http://localhost:8080/employee/Juan/Lopez | Forma de la petición: http://localhost:8080/employee?name=Juan&lastname=Lopez |
| <pre>@GetMapping("/employee/{name}/{lastname}") public Employee getEmployee(@PathVariable("name") String name @PathVariable("lastname") String lastName) { return new Employee(name, lastName); }</pre> | <pre>@GetMapping(path = "/employee/") public Student findEmployee(@RequestParam String name, @RequestParam String lastName) { return new Employee(name, lastName); }</pre> |

2 | Múltiples parámetros con el método POST

@PostMapping

La anotación @PostMapping se usa para mapear solicitudes HTTP POST en métodos de controlador. Es una alternativa de:

@RequestMapping (method = RequestMethod.POST)

Los métodos anotados con @PostMapping manejan las solicitudes HTTP POST que coinciden con una URI determinada.

Ejemplo: tenemos nuestra clase Employee y creamos un método que, por ejemplo, agrega un nuevo empleado.

```
@RestController
@RequestMapping("/")
public class HelloRestController {

    @PostMapping(path = "/employee")
    public void addEmployee(@RequestBody Employee employee) {
        //...
    }

}
```

Payload

Como parte de una solicitud POST o PUT se puede enviar una carga útil -o mejor conocida por su palabra en inglés *payload*- de datos al servidor en el cuerpo (*body*) de la solicitud.

El contenido del cuerpo puede ser cualquier objeto JSON válido. Por ejemplo:

```
{
  "firstname": "Charly"
  "lastname" : "Brown",
  "username" : "charlyb",
  "email"    : charlyb@digitalhouse.com"
}
```

@RequestBody

Se utiliza para vincular una solicitud HTTP (HTTP request) con un objeto en un parámetro en un método de nuestro controlador.

Esta anotación asigna el cuerpo de la solicitud HTTP a un objeto de dominio, lo que permite la deserialización automática del cuerpo de la solicitud HTTP entrante en un objeto Java.

Ejemplo: tenemos nuestra clase Employee y creamos un método que haga algo con él utilizando @RequestBody.

```
public class Employee{  
    private Long id;  
    private String name;  
    private String lastName;  
}
```

```
@RestController  
@RequestMapping("/")  
public class HelloRestController {  
  
    @PostMapping(path = "/employee")  
    public void handle(@RequestBody Employee  
employee) {  
        //...  
    }  
}
```

@ResponseBody

Es una anotación utilizada para indicar el contenido de una respuesta HTTP (*response*) dentro de su cuerpo.

Una respuesta HTTP no puede contener objetos Java, por lo que @ResponseBody se encarga de transformar los objetos a formato JSON o XML.

Ejemplo: el método `getOrders` devuelve una lista de órdenes. @ResponseBody se encarga de transformar esa lista de objetos en un json.

```
@GetMapping(path = "/orders/")
@ResponseBody
public List<Order> getOrders(){
    return orderService.getAllOrders();
}
```

DigitalHouse>