

Tabla de Contenidos



- [Spring @RequestParam](#)
- [Multiples Parámetros](#)
- [@RequestParam y defaultValues](#)
- [Otros artículos relacionados :](#)

Spring @RequestParam es una de las anotaciones más típicas cuando trabajamos con Spring o MVC ya sea a nivel de controladores como a nivel de servicios REST . Vamos a ver cuales son sus opciones a la hora de trabajar con ella . Para ello vamos a construir un ejemplo sencillo basado en la clase Persona.

CURSO SPRING FRAMEWORK
GRATIS
APUNTATE!!

```
package com.arquitecturajava.web;
```

```
public class Persona {
```

```
    private String nombre;  
    private String apellidos;  
    private int edad;
```

```
    public Persona(String nombre, String apellidos, int edad) {  
        super();  
        this.nombre = nombre;  
        this.apellidos = apellidos;
```

```
        this.edad = edad;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellidos() {
        return apellidos;
    }

    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

Vamos a construir con Spring Boot un sencillo RESTController .

```
package com.arquitecturajava.web;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;
import java.util.stream.Collectors;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class PersonaController {

    static List<Persona> lista= new ArrayList();
    static {
        lista.add(new Persona ("pepe","perez",20));
        lista.add(new Persona ("ana","gomez",30));
        lista.add(new Persona ("david","lopez",40));
    }
    @RequestMapping("/personas")
    public List<Persona> buscarTodos() {
        return lista;
    }
}
```

En este caso simplemente hemos utilizado la anotación @RequestMapping para mapear la url de /personas a una lista estática de personas que almacenamos en memoria.

Spring @RequestParam

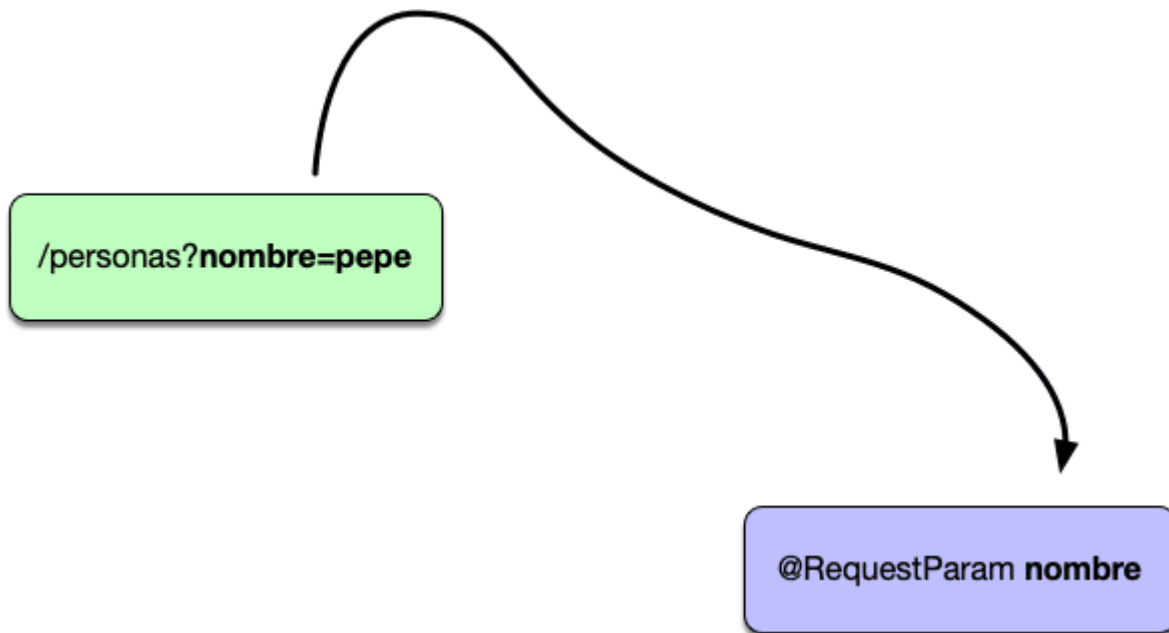
Una vez que hemos dado de alta el primer mapeo simplemente invocamos la url en el navegador y podremos ver que nos devuelve la lista de objetos en formato JSON.

```
[{"nombre":"pepe","apellidos":"perez","edad":20},
{"nombre":"ana","apellidos":"gomez","edad":30},
{"nombre":"david","apellidos":"lopez","edad":40}]
```

Es momento de añadir las urls que nos permitan filtrar a través del uso de parámetros :

```
@RequestMapping(value = "/personas" , params="nombre")
    public List<Persona> buscarTodosPorNombre(@RequestParam String
nombre ) {
    return
lista.stream().filter((p)->p.getNombre().equals(nombre)).collect(Colle
ctors.toList());
}
@RequestMapping(value = "/personas" , params="apellidos")
    public List<Persona> buscarTodosPorApellidos(@RequestParam String
apellidos ) {
    return
lista.stream().filter((p)->p.getApellidos().equals(apellidos)).collect
(Collectors.toList());
}
```

Acabamos de utilizar la anotación @RequestParam que es capaz de leer los parámetros que adjuntemos a la url:



Veamoslo en acción:

```
⬅ ➡ ↻ ⓘ localhost:8080/personas?nombre=pepe  
[{"nombre": "pepe", "apellidos": "perez", "edad": 20}]
```

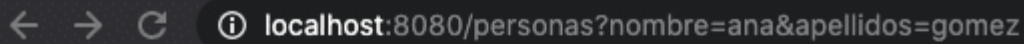
Multiples Parámetros

Esta anotación permite también el uso de multiples parámetros a nivel de la url ya que por ejemplo podemos consultar por nombre y apellidos añadiendo múltiples parámetros a nivel de las anotaciones:

```
@RequestMapping(value = "/personas" , params={"nombre","apellidos"} )  
public List<Persona> buscarTodosPorNombreyApellidos(@RequestParam  
String nombre ,@RequestParam String apellidos ) {  
    Predicate<Persona> filtro=  
(p)->p.getApellidos().equals(apellidos);  
    filtro.and((p->p.getNombre().equals(nombre)));  
    return lista.stream().filter(filtro).collect(Collectors.toList());  
}
```

```
}
```

El resultado será :



```
[{"nombre": "ana", "apellidos": "gomez", "edad": 30}]
```

@RequestParam y defaultValues

Hay situaciones en las cuales nos puede interesar tener valores por defecto para los parámetros en el caso de que no se envíen rellenos y se puedan procesar de forma natural:

```
@RequestMapping(value = "/personas" , params="orden")
public List<Persona> buscarTodos(@RequestParam (defaultValue =
"nombre" ) String orden ) {

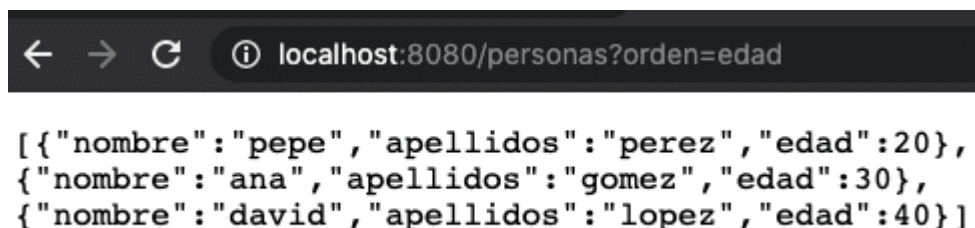
    if (orden.equals("nombre")) {
        return
lista.stream().sorted(Comparator.comparing(Persona::getNombre)).collect(
Collectors.toList());
    }else if (orden.equals("apellidos")) {
        return
lista.stream().sorted(Comparator.comparing(Persona::getApellidos)).collect(
Collectors.toList());
    }else {
        return
lista.stream().sorted(Comparator.comparing(Persona::getEdad)).collect(
Collectors.toList());
    }
}
```

De esta manera si nosotros queremos obtener una lista de personas ordenada será tan sencillo como :

```
@RequestMapping(value = "/personas" , params="orden")
public List<Persona> buscarTodos(@RequestParam (defaultValue =
"nombre" ) String orden ) {

    if (orden.equals("nombre")) {
        return
lista.stream().sorted(Comparator.comparing(Persona::getNombre)).collect(
Collectors.toList());
    }else if (orden.equals("apellidos")) {
        return
lista.stream().sorted(Comparator.comparing(Persona::getApellidos)).collect(
Collectors.toList());
    }else {
        return
lista.stream().sorted(Comparator.comparing(Persona::getEdad)).collect(
Collectors.toList());
    }
}
```

Vamos a verlo en ejecución :

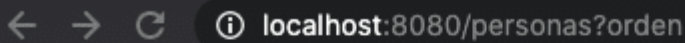


← → ↻ ⓘ localhost:8080/personas?orden=edad

```
[{"nombre":"pepe","apellidos":"perez","edad":20},
{"nombre":"ana","apellidos":"gomez","edad":30},
{"nombre":"david","apellidos":"lopez","edad":40}]
```

Acabamos de solicitar que nos ordene la lista por edad , pero también nos podemos encontrar que solicitemos la lista y se nos olvide pasar el parametro "orden" por ejemplo

alguien no lo relleno en el formulario . Al disponer de un parámetro por defecto no hay problema la lista vendrá ordenada por nombre:



← → ↻ ⓘ localhost:8080/personas?orden

```
[{"nombre": "ana", "apellidos": "gomez", "edad": 30},  
{ "nombre": "david", "apellidos": "lopez", "edad": 40},  
{ "nombre": "pepe", "apellidos": "perez", "edad": 20}]
```

El uso de @RequestParam es fundamental en el día a día de los desarrollos web.

CURSO SPRING REST
GRATIS
APUNTATE!!

Otros artículos relacionados :

1. [Spring MVC @RequestMapping](#)
2. [Spring REST Service con @RestController](#)
3. [¿Que es Spring Framework?](#)
4. [Spring MVC](#)