

# Spring Boot MVC

DigitalHouse>



**Certified Tech  
Developer**

The Ultimate Degree

## Definiendo un @Controller

La anotación @Controller indica que una clase particular cumple la función de un controlador.

El patrón Controlador sirve como intermediario entre una interfaz y el algoritmo que implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

## **El trabajo del controlador es:**

**Obtener los  
parámetros  
HTTP (si los hay).**

**Disparar la lógica  
de negocio.**

**Colocar el  
resultado en un  
ámbito accesible  
a la vista y  
cederle el  
control a este.**

**Pero... ¿Cómo hacemos para que, dada una petición en particular, un controlador determinado se encargue de entender este recurso y llamar a la vista correspondiente?**

Imaginemos que escribimos en el navegador:

`https://localhost:8080/hello`



# @RequestMapping

Utilizamos el `@RequestMapping` para relacionar una URL a una **clase** completa o a un **método** del controlador particular. Veamos un ejemplo:

```
@Controller
@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
}
```

# { Código }

Esta anotación define la clase HelloController como un controlador de Spring MVC.

Ruta que podemos introducir en el navegador. Por ejemplo:  
`http://localhost/hello`

```
@Controller
@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
}
```

Nombre de la vista, es decir, hello.html (el sufijo por defecto es html).

Información que enviamos a la plantilla, a la vista.

```
@Controller
@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Sp");
        return "hello";
    }
}
```

Usando  
@RequestMapping("/hello")  
asociamos una URL a la clase  
Hello controlador, es decir,  
indicamos que todos los  
métodos de manejo en este  
controlador son relativos a la  
ruta: "/hello".

<http://localhost:8080/hello>

```
@Controller
@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring");
        return "hello";
    }
}
```

Para que Spring sepa qué método del controlador debe procesar la petición HTTP, podemos especificar qué método HTTP asociaremos al método Java.

Entonces, si hacemos una llamada a la misma URL con POST produciría un error HTTP de tipo 404 porque no hay nada asociado a dicha petición.



## Otra forma:

```
@Controller
public class HelloController {
    @RequestMapping(value = "/hello", method = RequestMethod.GET)
    public String printHello(Model model) {
        model.addAttribute("message", "Hello Spring MVC
Framework!");
        return "hello";
    }
}
```

# Parámetros

Los parámetros de URL contienen una clave y un valor separados por un signo igual (=) y unidos por un signo de unión (&).

El primer parámetro siempre se ubica después del signo de interrogación en una URL. Por ejemplo:

`http://example.com/listaOfertas?mes=1&user=google`

## Obtener parámetros HTTP

```
@Controller
public class ListaOfertasController {

    @RequestMapping(value= "/listaOfertas", method= RequestMethod.GET)
    public String procesar(Model model, @RequestParam("mes") int mes) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        model.addAttribute("mes", mes);
        return "ofertas";
    }
}
```

**@RequestParam** asocia y convierte un parámetro HTTP a un parámetro Java.

## Obtener parámetros HTTP

Teniendo en cuenta nuestro ejemplo:

<http://example.com/listaOfertas?mes=1&user=google>



El parámetro mes tendrá el  
valor 1.

## Otras anotaciones

[@GetMapping](#), [@PostMapping](#), [@PutMapping](#), [@DeleteMapping](#) aparecen con Spring 4.3 y nos permiten simplificar el manejo de los diferentes métodos de Spring MVC y los `@RequestMapping`s que a veces se hacen un poco pesados.

```
@Controller
public class HelloController {
    @GetMapping("/hello")
    public String printHello(Model model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
    @PostMapping("/guardar")
    public String guardarProducto(@RequestBody Employee employee) {
        return "has hecho una peticion post";
    }
}
```

DigitalHouse>