

# CONTROLADORES

# ESTEREOTIPOS

- @Component
- @Service
- @Repository
- **@Controller**

# CONTROLADOR

- Clase POJO
- @Controller a nivel de clase

# ENDPOINT

- Asociar una URL + verbo a un método
  - @GetMapping, @PostMapping, ...

```
@GetMapping("/")  
public String welcome() {  
    return "index";  
}
```

# ESTRUCTURA DEL MÉTODO

Ruta del endpoint

```
@GetMapping("/")  
public String welcome() {  
    return "index";  
}
```

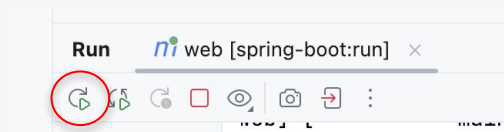
Nombre de la plantilla

No hay argumentos obligatorios

The diagram illustrates the structure of a Spring MVC controller method. It features a code snippet with three annotations: `@GetMapping` (yellow), `String` (blue), and `index` (green). Three arrows point from text labels to these annotations: 'Ruta del endpoint' points to `@GetMapping`, 'Nombre de la plantilla' points to `index`, and 'No hay argumentos obligatorios' points to the `String` type. The code snippet is enclosed in a light gray box.

# EJECUTAMOS PARA PROBAR

- ¡OJO! La ejecución **“no termina nunca”**
- Se levanta un servidor de Tomcat
- Para volver a ejecutar hay que parar la ejecución actual



# CÓMO ENVIAR DATOS A LA VISTA

## Contenedor



```
@GetMapping("/")  
public String welcome(Model model) {  
    model.addAttribute(attributeName: "message", attributeValue: "Hola Mundo!");  
    return "index";  
}
```

The diagram illustrates the process of passing data from a Spring controller to a Thymeleaf view. A straight arrow points from the `Contenedor` header to the `@GetMapping("/")` method in the controller. A wavy arrow points from the `message` attribute in the `addAttribute` call to the `th:text` attribute in the HTML template below.

```
<h1 th:text="${message}">Hello world!</h1>
```

# PREGUNTAS Y RESPUESTAS

- ¿Cuántos endpoint puede tener un controlador?
- ¿Cuántos datos se puede pasar al modelo?
- ¿Y si no quiero usar HTML?



# OTRAS FIRMAS PARA MÉTODOS DE CONTROLADOR

- ModelAndView

```
@GetMapping("/index") new *  
public ModelAndView welcomeV2() {  
    ModelAndView modelAndView = new ModelAndView( viewName: "index");  
    modelAndView.addObject( attributeName: "message", attributeValue: "Hola Mundo!");  
    return modelAndView;  
}
```

# OTRAS FIRMAS PARA MÉTODOS DE CONTROLADOR

- Tipos de retorno
  - *LoQueSea* + `@ResponseBody`
  - `ResponseEntity<?>`
  - `@ModelAttribute`
  - ...

# OTRAS FIRMAS PARA MÉTODOS DE CONTROLADOR

- Argumentos
  - @RequestParam, @PathVariable
  - @RequestBody
  - Errors, BindingResult
  - ...

# @REQUESTMAPPING

- Anotación *padre* de @GetMapping, @PostMapping
- Se puede usar a nivel de clase
- Permite indicar un tramo de ruta común a todos los métodos del controlador