Anotacion @RequestMapping – Nivel Método

- La anotación **@RequestMapping** cuando se incluye antes de la declaración de un método en un controlador sirve para especificar la URL y el Método HTTP (POST, GET, DELETE, PUT, etc) al que estará mapeado el método.
  - ✓El funcionamiento es similar al de las anotaciones **@GetMapping, @PostMapping,** etc.
- En versiones anteriores de **Spring Framework 4.3** se utilizaba esta anotación para mapear los métodos de los controladores a las URLs.
- Apartir de la versión **Spring Framework 4.3** se agregaron las siguientes variaciones de la anotación **@RequestMapping.**

Anotaciones (Spring 4.3+)	Mapeo con la anotación @RequestMapping	Uso común
@GetMapping("/lista")	@RequestMapping(value="/lista", method=RequestMethod.GET)	Desarrollo de aplicaciones web y RestFul WebServices.
@PostMapping("/guardar")	@RequestMapping(value="/guardar", method=RequestMethod.POST)	Desarrollo de aplicaciones web y RestFul WebServices.
@DeleteMapping("/borrar")	@RequestMapping(value="/borrar", method=RequestMethod.DELETE)	Desarrollo de RestFul WebServices.
@PutMapping("/actualizar")	@RequestMapping(value="/actualizar", method=RequestMethod.PUT)	Desarrollo de RestFul WebServices.

Anotación @RequestMapping – Ejemplo Nivel Método

```
@Controller
public class CategoriasController {

	// @GetMapping("/index")
	@RequestMapping(value="/index", method=RequestMethod.GET)
	public String mostrarIndex(Model model) {
		return "categorias/listCategorias";
	}

	// @GetMapping("/create")
	@RequestMapping(value="/create", method=RequestMethod.GET)
	public String crear() {
		return "categorias/formCategoria";
	}

	// @PostMapping("/save")
	@RequestMapping(value="/save", method=RequestMethod.POST)
	public String guardar() {
		return "categorias/listCategorias";
	}
}
```

Anotación @RequestMapping – A nivel de la Clase

```
@Controller
@RequestMapping(value="/categorias")
public class CategoriasController {

    @RequestMapping(value="/index", method=RequestMethod.GET)
    public String mostrarIndex(Model model) {
        return "categorias/listCategorias";
    }

    @RequestMapping(value="/save", method=RequestMethod.POST)
    public String guardar() {
        return "categorias/listCategorias";
    }
}
```

Al renderizarse la vista, la URL del atributo action del formulario HTML, se convierte a:

```
<form action="/categorias/save" >
```

Formulario HTML

```
<form th:action="@{/categorias/save}" method="post">
  Categoria
  <input type="text" name="nombre">
  <input type="submit" value="Guardar">
</form>
```

URL dinámicas

- Las URLs dinámicas (URI template) son usadas para obtener parte de ellas en un método de un controlador (como parámetro).
- Una URL dinámica puede contener 1 o varias PathVariable (parámetros entre llaves { }).
  - ✓ URL dinámica con un parámetro:
    - http://localhost:8080/detalle/**{id}**
      - **Se mandaría llamar como:** http://localhost:8080/detalle/**15**
  - ✓ URL dinámica con dos parámetros:
    - http://localhost:8080/detalle/**{id}**/**{fecha}**
      - **Se mandaría llamar como:** http://localhost:8080/detalle/**140**/**06-06-2019**
- Para vincular (binding) un parámetro de una URL dinámica a un parámetro en el controlador se utiliza la anotación @PathVariable:

```
@GetMapping("/detalle/{id}")
public String mostrarDetalle(@PathVariable("id") int idVacante){
  System.out.println("PathVariable: " + idVacante);
  return "detalle";
}
```

- Un método puede tener cualquier número de anotaciones @PathVariable

```
public String mostrarDetalle(@PathVariable("id") int idVacante, @PathVariable("fecha") Date fecha){
```

URL dinámicas – Ejemplo

Controlador

```
@GetMapping("/vacantes/view/{id}")
public String verDetalle(@PathVariable("id") int idVacante){
   System.out.println("PathVariable: " + idVacante);
   return "detalle";
}
```

Link HTML (GET)

```
<a th:href="@{/vacantes/view/{id} (id=${vacante.id}) }"> Ver detalles </a>
```

Generar URL dinámica

URL en el navegador (GET)

URL generada

Binding (vincular) Request Parameters

- Los parámetros de una petición HTTP pueden ser vinculados a un parámetro en un método en el controlador.
- Para vincular (binding) un parámetro de una petición HTTP se utiliza la anotación **@RequestParam** de Spring MVC. Se pueden vincular parámetros de una petición tipo GET y POST.

```
@GetMapping("/detalle")
public String verDetalle(@RequestParam("idVacante") int idVacante){
   // Procesamiento del parámetro. Aquí, ya se hizo la conversión a String a int.
   System.out.println("RequestParam: " + idVacante);
   return "someView";
}
```

- Los parámetros usados con esta anotación son **REQUERIDOS** por default.
- Si se require que el parámetro sea opcional, se tiene que agregar el atributo "required" con el valor "false".
  - ✓ (ejemplo, @RequestParam(name="id", **required=false**)).
- Por defecto los parámetros de la petición llegan al servidor de tipo String (HTTP protocol basado en texto). Sin embargo, Spring MVC tiene un componente de conversión que convierte de String al tipo indicado en el método (int, double, date, etc.).

Binding (vincular) Request Parameters (GET)

Controlador

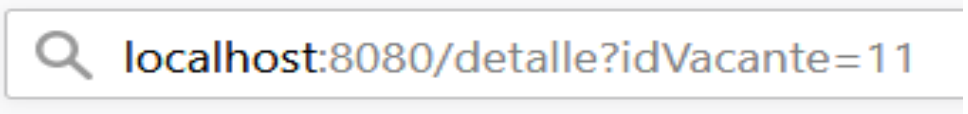
```
@GetMapping("/detalle")
public String verDetalle(@RequestParam("idVacante") int idVacante){
  // Procesamiento del parámetro. Aquí, ya se hizo la conversión a String a int.
  System.out.println("RequestParam: " + idVacante);
  return "someView";
}
```

Link HTML (GET)

```
<a th:href="@{/detalle(idVacante=${vacante.id}) }"> Ver detalles </a>
```

Esta expresión Thymeleaf en tiempo de ejecución se convierte a:
localhost:8080/detalle?idVacante=11

URL en el navegador (GET)



Binding (vincular) Request Parameters (POST)

FORM HTML (POST)

```
<form action="save" method="POST">
  Titulo
  <input type="text" name="titulo"/>
  <button type="submit" >Guardar</button>
</form>
```

Controlador

```
@PostMapping("/save")
public String guardar( @RequestParam("titulo") String tituloTmp  ){
   System.out.println("Titulo:" + tituloTmp);
   return "detalle";
}
```