

CONSTRUCCIÓN DE URLs



EXPRESIONES BÁSICAS

- ▶ Expresiones variables: `${...}`
- ▶ Expresiones variables de selección: `*{...}`
- ▶ Expresiones de mensaje: `#{...}`
- ▶ **Expresiones de enlaces: `@{...}`**
- ▶ Expresiones de fragmentos: `~{...}`

TIPOS DE **URLs**

- ▶ **Absolutas** (<https://openwebinars.net>)
- ▶ **Relativas**
 - ▷ **A la página** ([admin/categoria](#))
 - ▷ **Al contexto** ([/index?categoria=3](#))
 - ▷ Se añade el contexto automáticamente
 - ▷ **Al servidor** ([~/usuario/factura/](#))
 - ▷ Permite llamadas a URLs en otro contexto del mismo servidor
 - ▷ **Al protocolo** ([//code.jquery.com/jquery-3.3.1.min.js](#))
 - ▷ Añade el mismo protocolo que se está usando actualmente.

ETIQUETAS QUE USAN URLs

- ▶ `link`
- ▶ ``
- ▶ `<script src="..." th:src="@{...}" />`
- ▶ `<link href="..." th:href="..." />`

URLs CON PARÁMETROS O VARIABLES

- ▶ Las URLs pueden tener una parte llamada *query*, cuya información podemos utilizar en el controlador y la plantilla.
 - ▷ `/?idCategoria=5`
- ▶ También podemos usar *variables* como parte de la ruta de la URL
 - ▷ `/producto/10`

CONTROLADORES PARA URLs CON PARÁMETROS

- ▶ La gestión de los parámetros de la *query* de la URL la podemos hacer desde el controlador.

▶ `/?idCategoria=5`

Queremos que los productos que se muestren sean los de la categoría con

```
@GetMapping("/")
public String index(@RequestParam(name="idCategoria", required=false) Long idCategoria, Model model) {

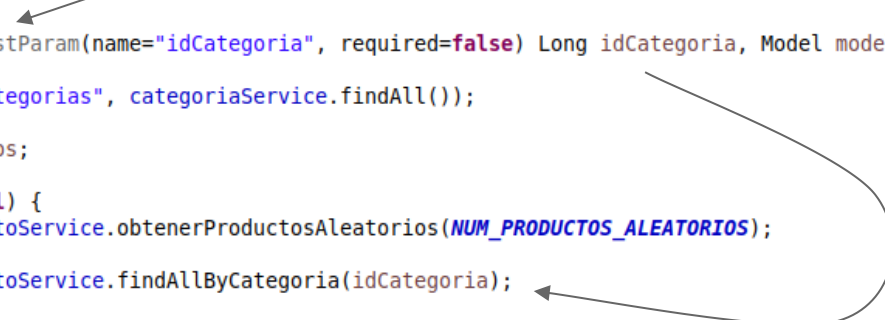
    model.addAttribute("categorias", categoriaService.findAll());

    List<Producto> productos;

    if (idCategoria == null) {
        productos = productoService.obtenerProductosAleatorios(NUM_PRODUCTOS_ALEATORIOS);
    } else {
        productos = productoService.findAllByCategoria(idCategoria);
    }

    model.addAttribute("productos", productos);

    return "index";
}
```



CÓMO CONSTRUIR URLs CON PARÁMETROS EN LA PLANTILLA

- ▶ En nuestra plantilla, necesitamos generar los enlaces necesarios.

Categorías

Portátiles	/?idCategoria=1
Componentes	/?idCategoria=2
Sobremesa	⋮
Smartphones	

```
<ul class="nav nav-sidebar">
  <li th:each="categoria : ${categorias}">
    <a href="#"
      th:href="@{/idCategoria=${categoria.id}}}"
      th:text="${categoria.nombre}">
      Categoria
    </a>
  </li>
</ul>
```

CÓMO CONSTRUIR URLs CON PARÁMETROS EN LA PLANTILLA

- Estructura genérica de una URL con parámetros

```
<a href="#"  
  th:href="@{/.../path?(p1=v1, p2=v2, p3=v3, ...)}"  
>...</a>
```

resultado

```
<a href="/.../path?p1=v1&p2=v2&p3=v3&...">...</a>
```


CÓMO CONSTRUIR URLs CON PARÁMETROS EN LA PLANTILLA

- ▶ Se pueden usar variables o literales como valores de los parámetros

```
<a href="#"  
  th:href="@{/?(idCategoria=${categoria.id},sort='PVP.ASC') }"  
>...</a>
```

resultado

```
<a href="/?idCategoria=1&sort=PVP.ASC">...</a>
```

CONTROLADORES PARA URLs CON VARIABLES EN EL PATH

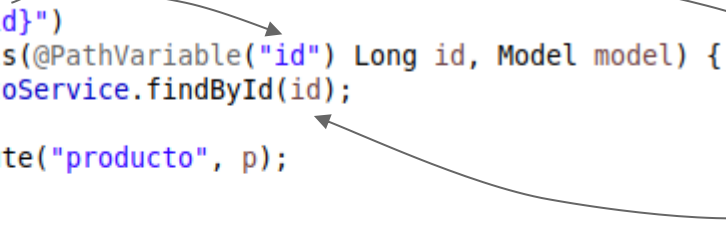
- ▶ La gestión de las variables en el *path* de la URL la podemos hacer desde el controlador.

- ▷ /product/17

← Queremos mostrar la información del producto con ID=17

```
@GetMapping("/product/{id}")
public String showDetails(@PathVariable("id") Long id, Model model) {
    Producto p = productoService.findById(id);
    if (p != null) {
        model.addAttribute("producto", p);
        return "detail";
    }

    return "redirect:/";
}
```



The diagram illustrates the flow of data from the URL to the controller and then to the model. An arrow points from the `{id}` in the `@GetMapping("/product/{id}")` annotation to the `id` parameter in the `showDetails` method signature. Another arrow points from the `id` parameter to the `findById(id)` call in the `productoService`. A third arrow points from the `producto` object returned by `findById` to the `addAttribute("producto", p)` call in the `if` block. Finally, a curved arrow points from the `return "detail";` statement back to the `addAttribute` call, indicating that the model is used to render the view.

CÓMO CONSTRUIR URLs CON VARIABLES EN EL PATH

- ▶ Estructura genérica de una URL con variables en el path sería esta

```
<a href="#"  
  th:href="@{/.../path/{var1}/{var2}(var1=v1, var2=v2)}"  
>...</a>
```

resultado

```
<a href="/.../path/v1/v2">...</a>
```

CÓMO CONSTRUIR URLs CON VARIABLES EN EL PATH

- ▶ En nuestra plantilla



/product/6

```
<a href="#"  
  th:href="@{/product/{id}(id=${producto.id})}">  
  <div class="col-item">  
    ...  
  </div>  
</a>
```