Tabla de Contenidos

El uso de Spring MVC @SessionAttributes nos ayuda a simplificar el manejo de estado a nivel de Spring MVC cuando necesitamos almacenar estado dentro de una sesión utilizando Spring MVC . La mayor parte de las veces en una aplicación web nos encontramos ante una navegación Stateless en la cual las páginas no mantienen estado . Sin embargo hay situaciones concretas como las de un carrito de la compra o acciones similares que nos vemos obligados a gestionar un estado a nivel de aplicación . Vamos a construir un ejemplo sencillo que nos ayude a entender mejor como funcionan las sesiones sobre Spring MVC. Para ello que mejor que el típico ejemplo de contador.

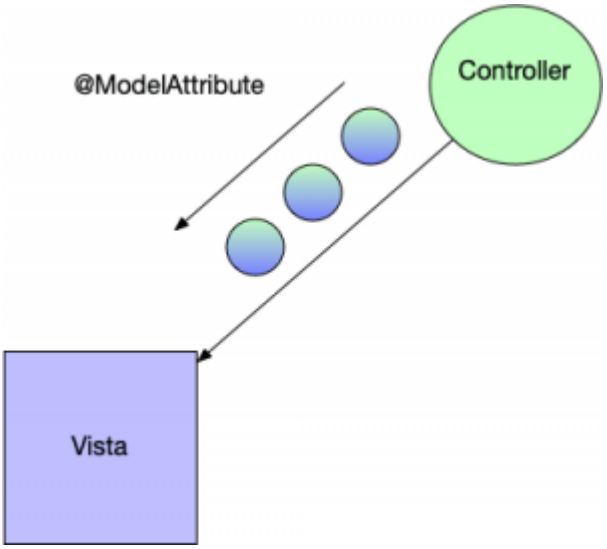
Acabamos de ver como usar Java equals y hashcode

```
package com.arquitecturajava;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.SessionAttributes;

@Controller

@SessionAttributes({"contador"})
public class ContadorController {

  @ModelAttribute("contador")
  public int getContador() {
```

```
    return 5;
  }
  @RequestMapping("/contador")
  public String verContador() {
    return "contador";
  }
  @RequestMapping("/contadorincrementar")
  public String incrementar(Model modelo) {
    int contador=(int) modelo.getAttribute("contador");
    contador++;
    modelo.addAttribute("contador",contador);
    return "contador";
  }
}
```

Spring MVC Controller

En este caso nos encontramos ante un Controlador que se encarga de mostrar un contador en una página a través de la anotacion @ModelAttribute . Recordar que esta anotación sirva para añadir una variable al objeto modelo con el cual nosotros intercambiamos información entre el controlador y la vista.



Esta anotación tiene como ciclo de vida la propia petición que nosotros realizamos al servidor por lo tanto cada vez que invoquemos la url de /contador nos mostrará el mismo resultado. Ahora bien echemos un vistazo a la plantilla . En ella se muestra claramente el contador como un pequeño formulario que permite invocar a la URL de /contadorincrementar.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<p th:text="${contador}"><p>
<form action="contadorincrementar">
<input type="submit"/>
</form>
</body>
</html>
```

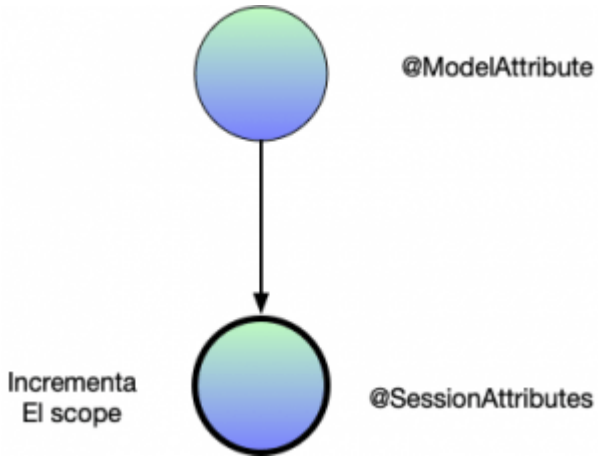
Esto hará que cuando se invoque la URL aparezca algo del siguiente estilo :

5

Enviar

Spring MVC @SessionAttributes

Cada vez que pulsemos en el boton de enviar se invocará la URL de /incrementarcontador que incrementará en uno el contador ya que hemos definido este @ModelAttribute como scope de Session al usar @SessionAttributes({“contador}”. Esto hará que el contador se almacene en sesión y cada vez que solicitemos la página se incremente.



Cada vez que pulsamos el botón de enviar el valor se incrementa en uno.

8

Enviar

Aprendamos a usar @SessionAttributes y @ModelAttributes en nuestras aplicaciones de Spring MVC

Otros artículos relacionados

1. @RequestParam , Spring MVC y sus opciones
2. Spring MVC Flash Attributes
3. JSF, Spring MVC y Java EE 8