Acabamos de ver como usar Java equals y hashcode

En el articulo anterior hemos configurado Spring MVC para cargar un ejemplo de Lista y Formulario .Ahora bien no hemos usado para nada el sistema de anotaciones de Spring . En este articulo cubriremos ese hueco. Lo primero que vamos a hacer es añadir un ViewResolver a nuestro fichero application-context.xml .

```
&amp;lt;bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolv
er"&amp;gt;
&amp;lt;property name="prefix" value="/WEB-INF/jsp/" /&amp;gt;
&amp;lt;property name="suffix" value=".jsp" /&amp;gt;
&amp;lt;/bean&amp;gt;
```

Se trata de un bean sencillo que parametriza la URL de las distintas vistas que cargamos. A partir de este momento nuestros controladores podran devolver algo como lo siguiente.

```
return new ModelAndView("Lista");
```

Spring MVC y @Controller

A partir de Spring 2.5 se permite el uso de un sistema de anotaciones que simplifica la forma de trabajar . Para usar este sistema deberemos en primer lugar añadir una nueva dependencia a nuestro pom.xml.

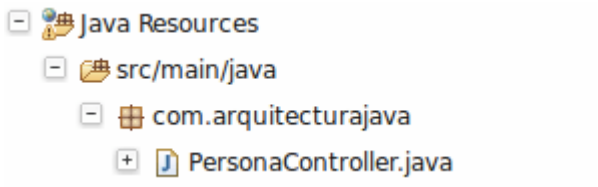
```
&amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
&amp;lt;artifactId&amp;gt;spring-context&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;3.2.4.RELEASE&amp;lt;/version&amp;gt;
```

```
&amp;lt;/dependency&amp;gt;
```

Esta dependencia nos permitirá modidificar el fichero application-context.xml y añadir una etiqueta

```
&amp;lt;context:component-scan base-
package="com.arquitecturajava"/&amp;gt;
```

Esta etiqueta se encarga de decirle a Spring en que packages debe buscar clases anotadas para su registro . En nuestro caso en “com.arquitecturajava” .Realizada esta operación creamos una clase PersonaController dentro del package.

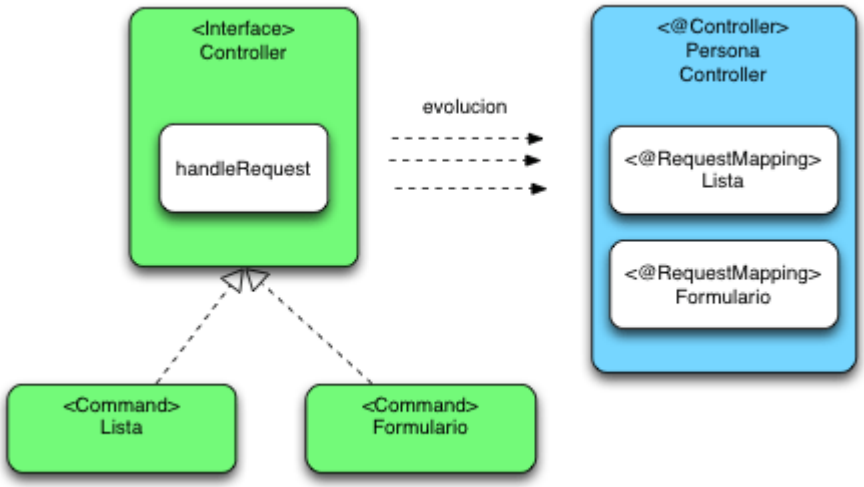


Vamos a ver su código fuente :

```
package com.arquitecturajava;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class PersonaController {
@RequestMapping("/Lista")
public String Lista() {
return "Lista";
}
@RequestMapping("/Formulario")
public String Formulario() {
return "Formulario";
```

```
}
}
```

Como podemos observar hemos evolucionado nuestro ejemplo y en vez de tener varios comandos . Disponemos de un controlador que agrupa un conjunto de acciones .

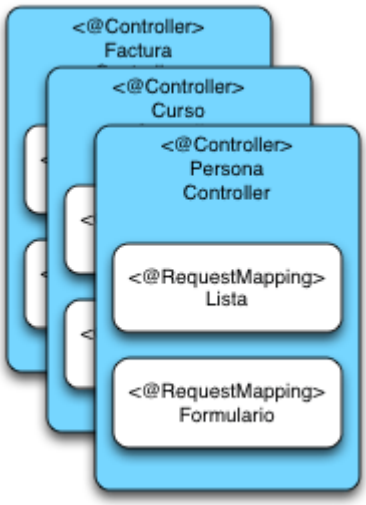


Vamos a explicar para que sirven las anotaciones que hemos utilizado.

@Controller: Anotación que registra el controlador para Spring MVC

@RequestMapping: Anotación que se encarga de relacionar un método con una petición http.

El uso de anotaciones no implica que unicamente tengamos un único controlador sino que nos permite agrupar un conjunto de urls que esten asociadas a nivel de negocio en un controlador especifico. La aplicación soportará n controladores.



Spring MVC y el Modelo

Por ahora lo único que hemos hecho es construir el controlador .Ahora bien el controlador suele enviar datos a la vista a traves de un modelo. Vamos a modificar PersonaController para que tenga una lista de Personas y nos las envie a Lista.jsp. El primer paso será añadir la clase Persona.

```
package com.arquitecturajava;
public class Persona {
private String nombre;
private String apellidos;
public Persona(String nombre, String apellidos) {
super();
this.nombre = nombre;
this.apellidos = apellidos;
}
//omitimos equals y hashcode eclipse

public String getNombre() {
return nombre;
```

```
}
public void setNombre(String nombre) {
this.nombre = nombre;
}
public String getApellidos() {
return apellidos;
}
public void setApellidos(String apellidos) {
this.apellidos = apellidos;
}
}
```

Realizada esta operación el siguiente paso será modificar el controlador para que genere una lista de Personas .

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class PersonaController {

@RequestMapping("/Lista")
public String Lista(Model modelo) {
List&amp;lt;Persona&amp;gt; milista= new
```

```
ArrayList&amp;lt;Persona&amp;gt;();
Persona persona1= new Persona("pedro","perez");
Persona persona2=new Persona ("maria","gonzalez");
milista.add(persona1);
milista.add(persona2);

modelo.addAttribute("listaPersonas",milista);
return "Lista";
}
@RequestMapping("/Formulario")
public String Formulario() {
return "Formulario";
}
}
```

Como podemos observar ahora el método lista soporta un objeto de tipo Model .Es en esto objeto en el cual nosotros añadimos la lista. Realizado este paso nos quedará modificar la pagina JSP para que nos muestre una lista utilizando JSTL. Para ello antes deberemos añadir la dependencia a Maven.

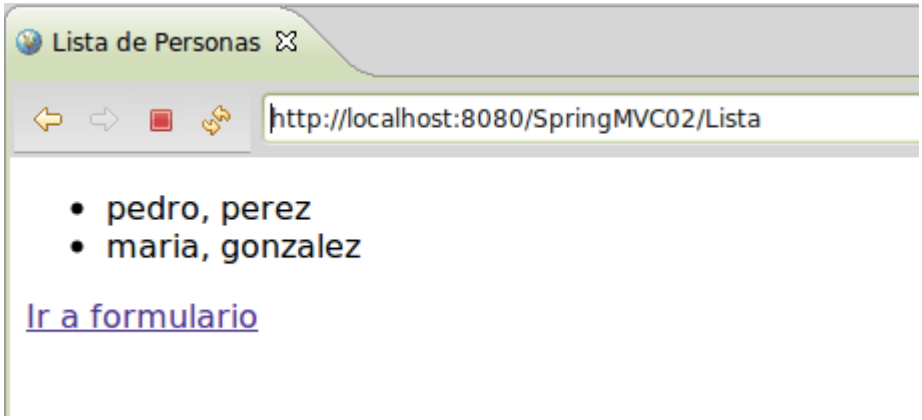
```
&amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;jstl&amp;lt;/groupId&amp;gt;
&amp;lt;artifactId&amp;gt;jstl&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;1.2&amp;lt;/version&amp;gt;
&amp;lt;/dependency&amp;gt;
```

Finalmente mostramos la página JSP que se encarga de mostrar la lista de personas.

```
&amp;lt;%@ page language="java" contentType="text/html; charset=UTF-8"
```

```
pageEncoding="UTF-8"%&amp;gt;
&amp;lt;%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%&amp;gt;
&amp;lt;!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd"&amp;gt;
&amp;lt;html&amp;gt;
&amp;lt;head&amp;gt;
&amp;lt;meta http-equiv="Content-Type" content="text/html;
charset=UTF-8"&amp;gt;
&amp;lt;title&amp;gt;Lista de Personas&amp;lt;/title&amp;gt;
&amp;lt;/head&amp;gt;
&amp;lt;body&amp;gt;
&amp;lt;ul&amp;gt;
&amp;lt;c:forEach items="${listaPersonas}" var="persona"&amp;gt;
&amp;lt;li&amp;gt;${persona.nombre},
${persona.apellidos}&amp;lt;/li&amp;gt;
&amp;lt;/c:forEach&amp;gt;
&amp;lt;/ul&amp;gt;
&amp;lt;a href="Formulario"&amp;gt;Ir a formulario&amp;lt;/a&amp;gt;
&amp;lt;/body&amp;gt;
&amp;lt;/html&amp;gt;
```

Para finalizar cargamos la página en un navegador y nos mostrará la lista.



Ya tenemos un ejemplo sencillo con anotaciones y Spring MVC funcionando