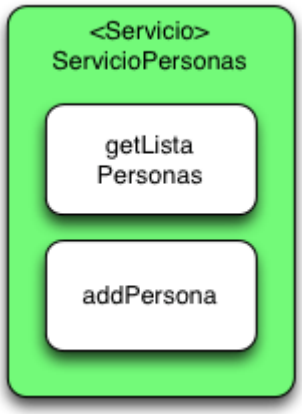
Acabamos de ver como usar Java equals y hashcode

En el post anterior de Spring framework MVC hemos visto como hacer uso de anotaciones para cargar una lista y un formulario . En este post modificaremos PersonaController para que admita los datos enviados desde un formulario y seamos capaces de añadir contenido a la lista. Lo primero que vamos a hacer es registrar una clase de servicio que permita estas operaciones a través de un ArrayList.

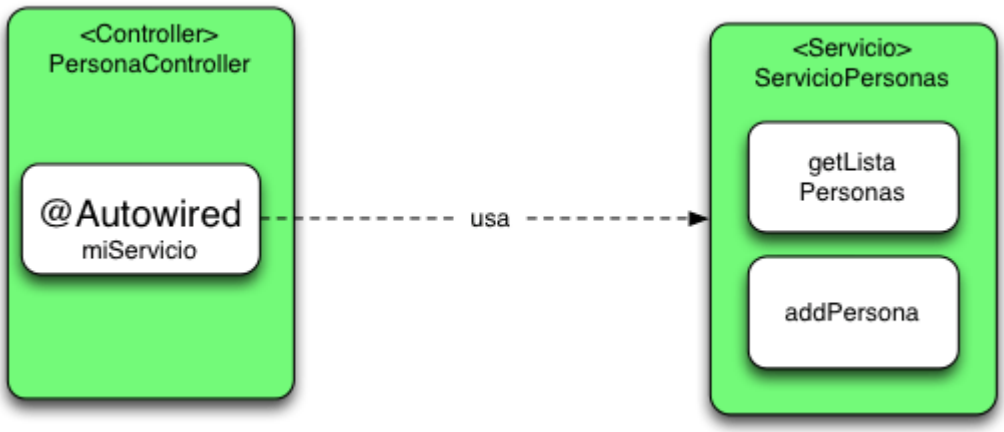


Mostramos a continuación su código:

```
package com.arquitecturajava;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Service;

@Service
public class ServicioPersona {

private List&amp;lt;Persona&amp;gt; listaPersonas= new
```

```
ArrayList&amp;lt;Persona&amp;gt;();

public List&amp;lt;Persona&amp;gt; getListaPersonas() {
 return listaPersonas;
 }

public void setListaPersonas(List&amp;lt;Persona&amp;gt;
listaPersonas) {
 this.listaPersonas = listaPersonas;
 }
 public void addPersona(Persona p) {

 listaPersonas.add(p);
 }
}
```

Una vez tenemos la clase de servicio deberemos modificar el controlador para que haga uso de ella utilizando la anotación @Autowired que nos permite inyectar de forma transparente la dependencia.



```
public class PersonaController {

@Autowired
 private ServicioPersona miservicio;

 @RequestMapping("/Lista")
 public String Lista(Model modelo) {

 modelo.addAttribute("listaPersonas",miservicio.getListaPersonas());
 return "Lista";
 }

@RequestMapping("/Formulario")
 public String Formulario() {

 return "Formulario";
 }
```

El siguiente paso será modificar el formulario para que invoque a una url que inserte los datos de la persona.

```
&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;form action="Insertar"&amp;gt;
Nombre&amp;lt;input type="text" name="nombre"
```

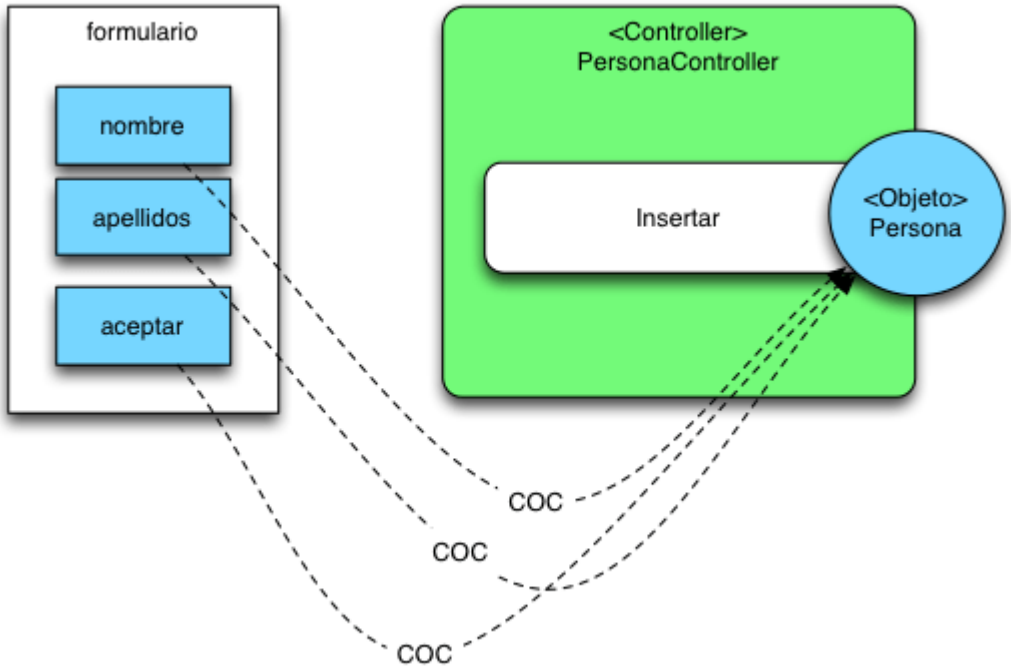
```
/&amp;gt;&amp;lt;/br&amp;gt;
Apellidos&amp;lt;input type="text" name="apellidos"
/&amp;gt;&amp;lt;/br&amp;gt;
&amp;lt;input type="submit" name="aceptar" value="Aceptar"/&amp;gt;
&amp;lt;/form&amp;gt;
&amp;lt;/body&amp;gt;
&amp;lt;/html&amp;gt;
&amp;lt;pre&amp;gt;
```

En este caso unicamente modificamos la acción y la asociamos a una nueva URL “Insertar” . Así pues deberemos añadir al controlador un nuevo método para que soporte la nueva URL.

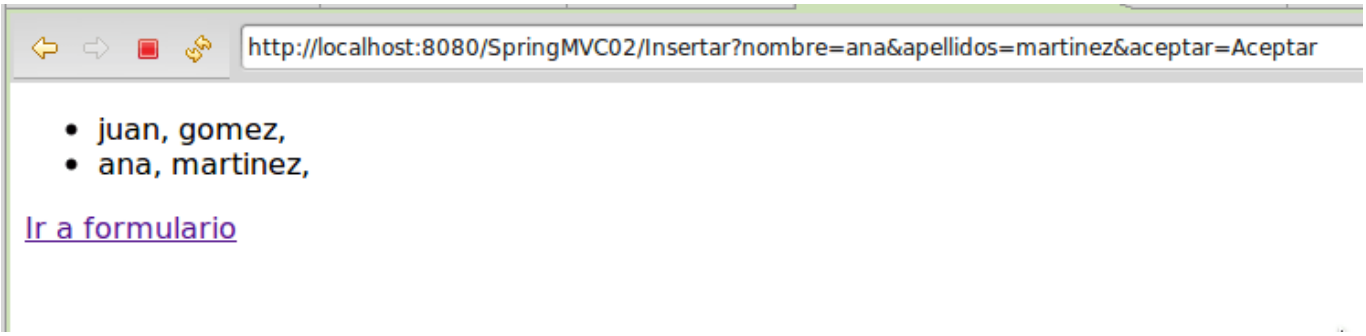
```
@RequestMapping("/Insertar")
 public String Insertar(Persona persona) {
 miservicio.addPersona(persona);

 return "forward:Lista";
 }
```

Convención sobre Configuración (COC)

El método Insertar recibe como parámetro un objeto de tipo Persona el cual será rellenado con los valores que vienen del formulario ya que Spring por convención mapea automaticamente los campos que tienen el mismo nombre en el formulario que en las propiedades de la clase . Permitiendonos acceder de forma directa al objeto que ya se encuentra relleno.



Por último solicitamos al método que realice un forward y vuelva a cargar la lista de registros.



Spring y Singletons

Si revisamos el código nos percataremos que estamos inyectando la clase ServicioPersonas y luego invocamos una y otra vez desde el formulario al método addPersona de esta clase. Recordemos que por defecto en Spring Framework todos los beans que se instancian son Singletons. Es decir unicamente existe una instancia de la clase que es compartida por todos.