Acabamos de ver como usar Java equals y hashcode

Crear un Spring REST Service es ahora muy sencillo a través del uso de la anotación @RestController que Spring 4 soporta. En primer lugar se configura el fichero de pom.xml de Maven con las siguientes dependencias.

```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>4.1.7.RELEASE</version>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>servlet-api</artifactId>
<version>2.5</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-validator</artifactId>
<version>5.1.3.Final</version>
</dependency>

<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
```

```
<artifactId>jackson-core</artifactId>
<version>2.4.2</version>
</dependency>

<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.4.2</version>
</dependency>
</dependencies>
```

El siguiente paso es configurar el Servlet Dispatcher en el web.xml que es el que nos permite mapear todas las urls de los controladores de Spring.

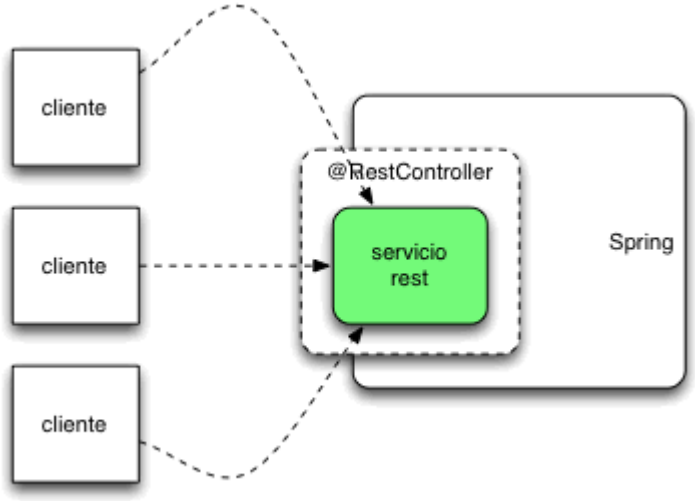
```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">

<servlet>
<servlet-name>ServletSpring</servlet-name>
<servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
<init-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/config/applicationContext.xml</param-value>
</init-param>
</servlet>

<servlet-mapping>
```

```
<servlet-name>ServletSpring</servlet-name>
<url-pattern>*.html</url-pattern>
<url-pattern>*.json</url-pattern>
</servlet-mapping>

</web-app>
```

El último paso a nivel de configuración es definir el fichero applicationContext.xml:

```
<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
http://www.springframework.org/schema/mvc12
http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">
<mvc:annotation-driven />
<context:component-scan base-package="com.arquitecturajava.controller"
/>
</beans>
```

Realizada la configuración será suficiente con crear una clase que use la anotación @RestController y automaticamente se publicara como un Spring REST Service.

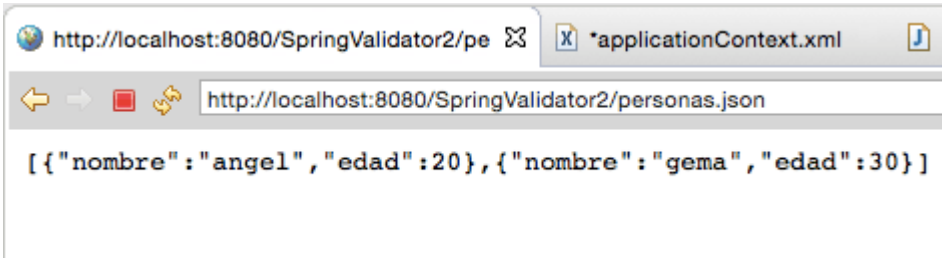



```
package com.arquitecturajava.controller;

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

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.arquitecturajava.negocio.Persona;

@RestController
public class ControladorREST {

 @RequestMapping(value = "/personas", method = RequestMethod.GET)
 public List < Persona > listaPersonas() {

  List < Persona > lista = new ArrayList < Persona > ();
  Persona p = new Persona();
  p.setNombre("angel");
```

```
  p.setEdad(20);
  lista.add(p);
  Persona p1 = new Persona();
  p1.setNombre("gema");
  p1.setEdad(30);
  lista.add(p1);
  return lista;
 }
}
```

Acabamos de crear un servicio REST que nos devuelve una Persona en una url determinada , concretamente en /personas recordemos que para las url REST normalmente se utilizan los plurales.

Ya solo queda realizar una petición web a la URL en donde se ha mapeado el Spring REST Service:



Conclusiones

Spring 4

introduce novedades que hacen la vida más sencilla a los desarrolladores

Otros artículos relacionados:

- Introducción a Spring MVC
- Spring MVC Bean Validation