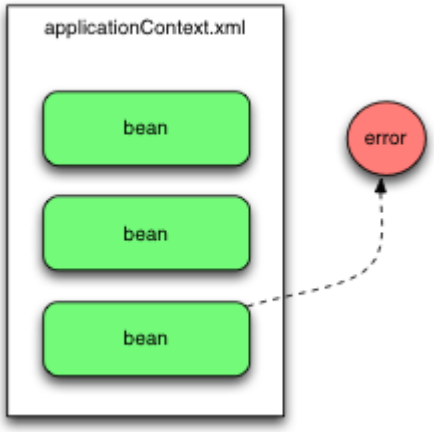
Acabamos de ver como usar Java equals y hashcode

Spring es uno de los frameworks mas habituales hoy en día a la hora de trabajar en JEE. Si tuviera que revisar los puntos débiles que el framework tiene , destacaría el tener que leer el fichero applicationContext.xml al iniciarle y los problemas de ClassNotFoundException que se suelen producir ya sea porque nos faltan librerias o porque hemos escrito incorrectamente el nombre de las clases.



Vamos a suponer que disponemos del siguiente fichero de configuración (applicationContext.xml).

applicationContext.xml

```
&amp;lt;?xml version="1.0"
encoding="UTF-8"?&amp;gt;&amp;lt;/pre&amp;gt;
&amp;lt;beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans.xsd"&gt;

&amp;lt;bean id="servicioA"
```

```
 class="com.arquitecturajava.ServicioA"&amp;gt;
 &amp;lt;/bean&amp;gt;

&amp;lt;bean id="servicioB"
 class="com.arquitecturajava.ServicioB"&amp;gt;
 &amp;lt;/bean&amp;gt;
&amp;lt;/beans&amp;gt;
&amp;lt;pre&amp;gt;
```

Como podemos ver hemos definido dos beans en el fichero servicioA y servicioB .Podemos usarlos en un programa de consola de la siguiente forma.

Principal.java

```
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;

public class Principal {

 public static void main(String args[]) {

 ApplicationContext contexto = new
ClassPathXmlApplicationContext("applicationContext.xml");
 ServicioA servicioA= (ServicioA) contexto.getBean("servicioA");
 System.out.println(servicioA.mensaje());

 }
```

```
}
```

Aunque el uso es correcto siempre aparecen problemas como por ejemplo que en el método getBean hayamos escrito ServicioA con la S “Mayuscula” o que en el fichero applicationContext.xml no esten bien puestos los nombres de las clases etc .Son errores sencillos de corregir pero son errores molestos. Vamos a ver como eliminarlos.

Spring @configuration

Spring soporta anotaciones de configuración a partir de su versión 3.0 que nos permiten enfocar de otra forma . Vamos a declarar una clase “ConfiguracionApp” que se encargue de realizar la configuración de la aplicación sin usar el fichero “applicationContext.xml” que es el que nos esta dando problemas .

```
package com.arquitecturajava;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ConfiguracionApp {

@Bean
 public ServicioA servicioA() {

 return new ServicioA();
 }
 @Bean
 public ServicioB servicioB() {
```

```
 return new ServicioB();
 }
}
```

Si revisamos la clase ConfiguracionApp dispone de dos anotaciones de Spring Framemork .

Spring @Configuration :Anotación encargada de definir que la clase es una clase de configuración para el framework
Spring @Bean :Anotación que marca como bean cada uno de los métodos de tal forma que esten disponibles para Spring

Realizadas estas modificaciones nuestra aplicación no necesitará utilizar ya el fichero applicationContext.xml . Todo quedará registrado con nuestra nueva clase que a nivel de compilación realizará las comprobaciones de nombres necesarias.Para usar la nueva clase de configuración con Spring en un programa de consola será tan sencillo como lo siguiente.

```
package com.arquitecturajava;

import
org.springframework.context.annotation.AnnotationConfigApplicationCont
ext;

public class PrincipalB {

 public static void main(String args[]) {

 AnnotationConfigApplicationContext contexto=
 new AnnotationConfigApplicationContext();
```

```
 contexto.register(ConfiguracionApp.class);
 contexto.refresh();

 ServicioA servicioA= (ServicioA) contexto.getBean(ServicioA.class);
 System.out.println(servicioA.mensaje());

 }

}
```

Podemos apreciar que ya no hemos usado en ningún momento Strings a la hora de invocar los diferentes “beans” registrados y podemos eliminar el fichero applicationContext.xml.

