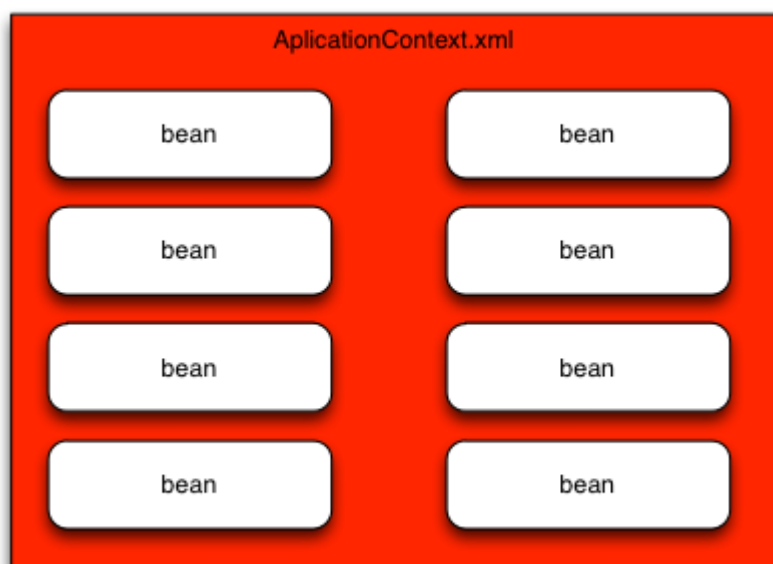
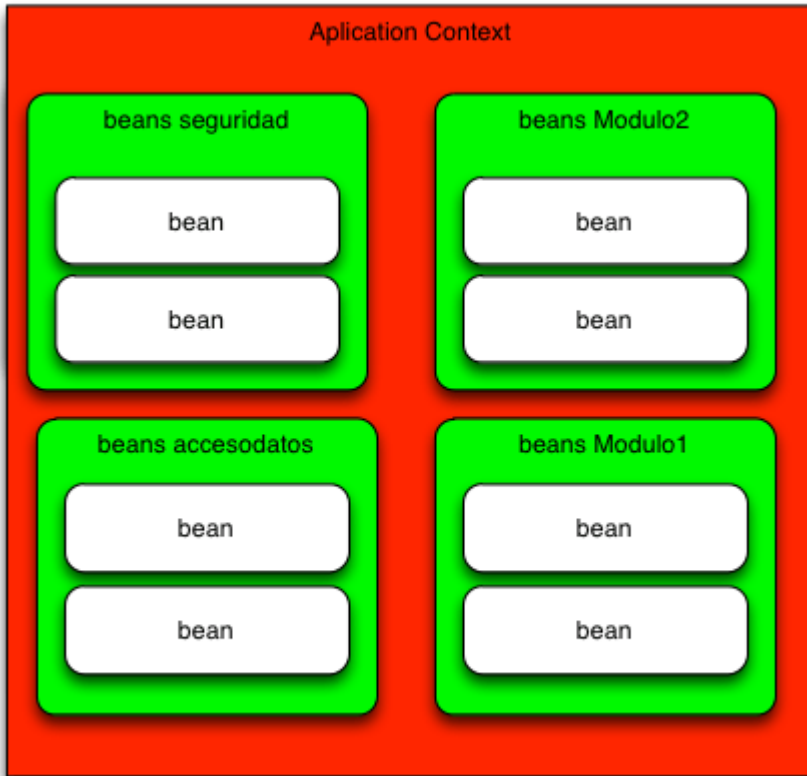


Acabamos de ver como usar Java equals y hashCode

En muchas ocasiones cuando trabajamos con Spring nos encontramos que el fichero de applicationContext.xml ha crecido de tamaño y estamos perdiendo el control sobre el . Tiene dado de alta un conjunto enorme de beans unas son servicios , otras DAO , otras temas de seguridad etc.



Este problema normalmente se puede mitigar con el uso de anotaciones (@Service,@Repository) etc . Sin embargo siempre hay proyectos que necesitan mas flexibilidad ya que tienen muchos beans dados de alta. Es en estos proyectos en los que poder organizar los beans y agruparlos por responsabilidad se convierte en algo obligatorio si queremos sobrevivir.



Vamos a mostrar un ejemplo sencillo de como organizar estos beans . Supongamos que nuestro fichero de applicationContext.xml tiene la siguiente estructura.

```
&lt;?xml version="1.0" encoding="UTF-8"?>
&lt;beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.com/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.2.xsd"&
  gt;
&lt;bean id="servicioModulo1"
  class="com.arquitecturajava.contexto.ServicioModulo1"&gt;
```

```
&lt;/bean>
&lt;/beans>
```

En este caso tenemos dos clases de servicio ServicioModulo1y ServicioModulo2 .Aunque nuestro ejemplo es trivial y no necesitaría partir el fichero vamos a hacerlo para clarificar el concepto . Para ello nos vamos a crear dos nuevos ficheros modulo1.xml y modulo2.xml . Cada uno de los cuales contendrá uno de los beans.

Modulo1.xml

```
&lt;?xml version="1.0" encoding="UTF-8"?>
&lt;beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.com/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.2.xsd"&
  gt;
  &lt;bean id="servicioModulo1"
    class="com.arquitecturajava.contexto.ServicioModulo1"&gt;
  &lt;/bean>
&lt;/beans>
```

Modulo2.xml

```
&lt;/pre>
&lt;?xml version="1.0" encoding="UTF-8"?>
&lt;beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.com/schema/context"
```

```

xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd"&
gt;
&lt;bean id="servicioModulo2"
class="com.arquitecturajava.contexto.ServicioModulo2"&gt;
&lt;/bean&gt;
&lt;/beans&gt;
&lt;/pre&gt;

```

Una vez que hemos configurado ambos ficheros tenemos que configurar el fichero principal de applicationContext.xml. Para ello usaremos las etiquetas <import> de Spring y añadiremos ambos módulos.

applicationContext.xml

```

&lt;/pre&gt;
&lt;?xml version="1.0" encoding="UTF-8"?&gt;
&lt;beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.com/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd"&
gt;
&lt;import resource="modulo1.xml"/&gt;
&lt;import resource="modulo2.xml"/&gt;

```

```
</beans>
<pre>
```

Realizadas estas modificaciones en nuestros ficheros de configuración el diseño será mas flexible y podremos organizarnos mejor.

