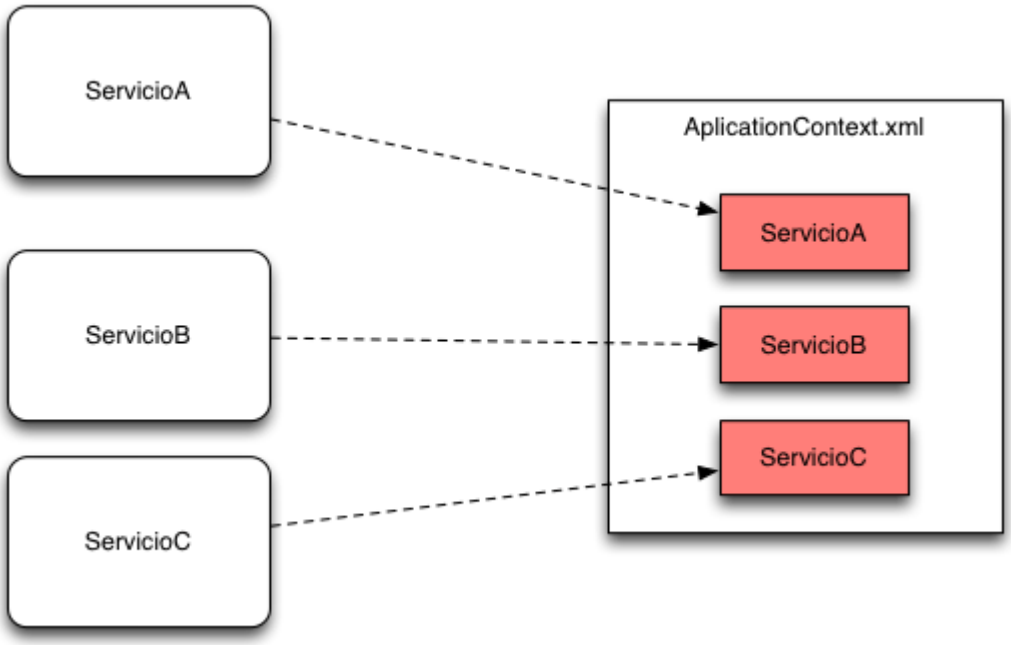
Acabamos de ver como usar Java equals y hashcode

Muchas veces la gente me pregunta si es mejor usar Anotaciones o ficheros XML a nivel de framework Spring . En un primer momento es más que evidente que el uso de ficheros XML hace que a veces la propia aplicación sea un poco inmanejable. Esto es debido a que cada uno de los Beans que utilizamos debe registrarse en el fichero XML.



Cuantos más beans tengamos más inmanejable será el fichero ApplicatiónContext.xml .Imaginemonos que tenemos los siguientes Beans (ServicioA,ServicioB) a continuación se muestra el fichero ApplicationContext.xml.

```
&amp;lt;beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
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.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;
```

Acabamos de registrar dos Servicios y vamos a mostrar su código:

```
package com.arquitecturajava;

public class ServicioA {

public String mensaje() {

return "hola servicio A";
}
}
```

```
package com.arquitecturajava;
```

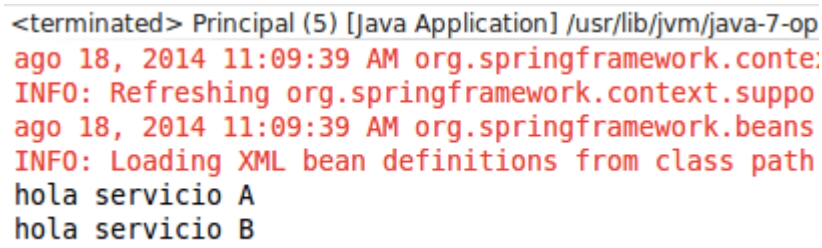
```
public class ServicioB {

public String mensaje() {

return "hola servicio B";
}
}
```

Como podemos ver ambos servicios simplemente devuelven un mensaje de “hola servicio X”. En estos momentos podemos configurar un programa sencillo con Spring para hacer uso de ellos.

```
package com.arquitecturajava;

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());
```

```
 ServicioB servicioB= (ServicioB) contexto.getBean("servicioB");
 System.out.println(servicioB.mensaje());

 }

}
```

El programa imprimirá por pantalla lo siguiente :



Sin embargo tenemos el problema de que cada vez registraremos más y más servicios a nivel del fichero XML. Si queremos evitar esta situación podemos hacer uso de Anotaciones . En este caso de la anotación @Service que declararemos en cada una de nuestras clases servicio.

Vamos a verlo en código:

```
package com.arquitecturajava;

import org.springframework.stereotype.Service;

@Service
public class ServicioA {

 public String mensaje() {

 return "hola servicio A";
 }
}
```

```
package com.arquitecturajava;

import org.springframework.stereotype.Service;

@Service
public class ServicioB {

 public String mensaje() {

 return "hola servicio B";
 }
}
```

Realizada esta operación modificaremos el fichero XML para que cargue automaticamente todas las clases anotadas:

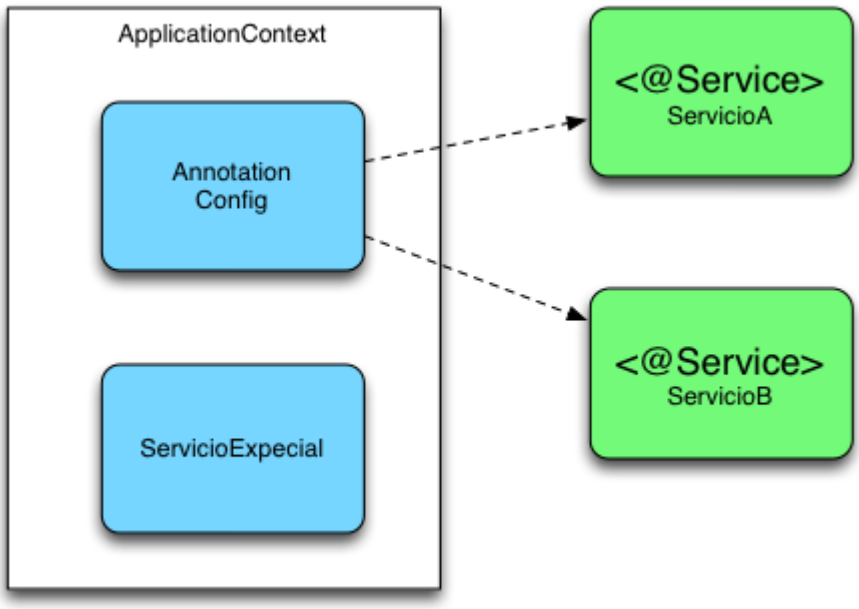
```
&amp;lt;beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd"&gt;

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

```
&amp;lt;/beans&amp;gt;
```

Para ello se usan dos etiquetas especiales “annotation-config” que se encarga de registrar todos los servicios anotados con @Service y component-scan que nos indica que package comprobar. Realizada esta operación el fichero XML queda practicamente vacío aunque existan cientos de servicios.

Spring anotaciones y flexibilidad

En un principio parece que es mucho mejor usar Anotaciones ya que nos ahorramos mucho código. Sin embargo a veces hay que dar una vuelta a nuestras conclusiones. Por ejemplo puede haber casos en que tengamos una serie de Servicios que siempre queramos trabajar con ellos de forma muy personalizada ya que han sido problemáticos o tenemos que tener en cuanta cosas a la hora de realizar migraciones. En este caso podemos realizar un enfoque mixto algo que mucha gente desconoce y tener una gran parte de los servicios con anotaciones y una pequeña parte a nivel de ficheros XML para recordarnos que eran especiales.



De esta forma Spring permite una gran flexibilidad a la hora de tratar los beans que manejamos .Recordemos ademas que Spring permite partir el fichero xml de configuración en varios. En nuestro caso podemos acabar configurando un servicio dentro de ApplicationContext.xml y el otro a traves de annotation-config.

```
&amp;lt;beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
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.xsd"&gt;

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

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

&amp;lt;/beans&amp;gt;
```

Otros Artículos relacionados:

SpringPropertyPlaceConfigurer

Spring Security