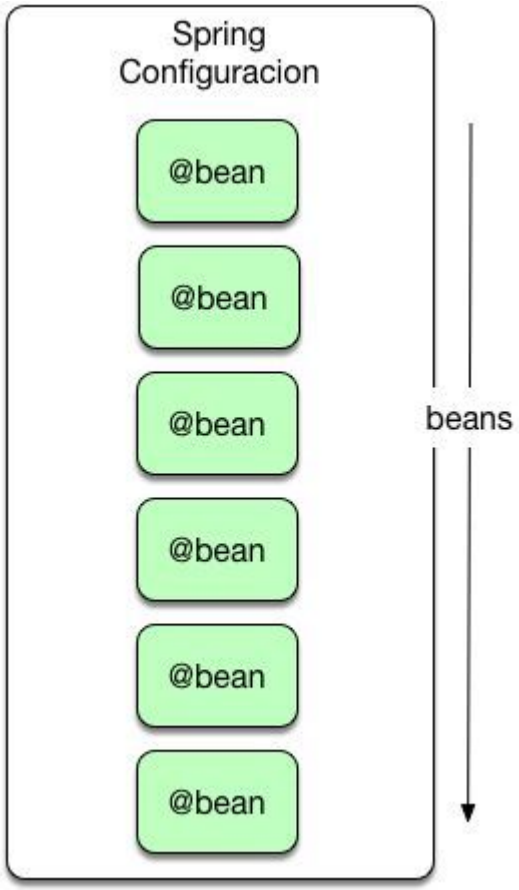
Acabamos de ver como usar Java equals y hashcode

Spring @import nos permite organizar de una mejor manera la configuración de Spring Framework . Normalmente en una configuración de Spring basada en anotaciones tenemos algo similar a lo siguiente:

```
package com.arquitecturajava.spring;

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

@Configuration
public class SpringConfiguracion {

	@Bean
	public static HelperPersistencia helper1() {

		return new HelperPersistencia();
	}

	@Bean
	public static HelperSeguridad helper2() {

		return new HelperSeguridad();
	}

}
```

En este caso el fichero es muy pequeño pero es bastante habitual que tenga cientos de lineas de código registrando cada uno de los beans que son necesarios , seguridad ,persistencia etc.



Podemos crear un programa principal y ejecutarlo:

```
package com.arquitecturajava.spring;

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

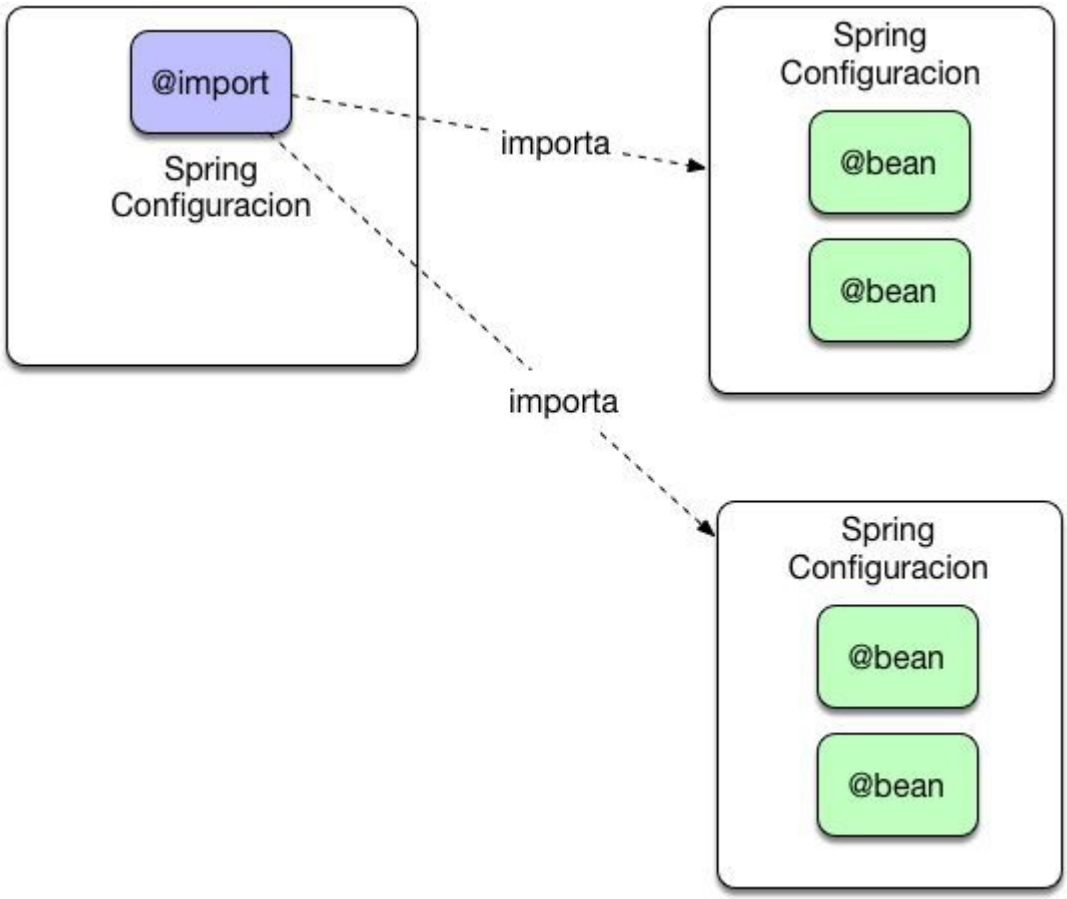
```
public class Principal {

	public static void main(String[] args) {

		AnnotationConfigApplicationContext contexto = new
AnnotationConfigApplicationContext();
		contexto.register(SpringConfiguracion.class);
		contexto.refresh();

		HelperSeguridad
helper1=contexto.getBean(HelperSeguridad.class);
		helper1.hola();
		HelperPersistencia
helper2=contexto.getBean(HelperPersistencia.class);
		helper2.hola();

	}

}
```

Veremos el resultado por la consola:

Los beans se ejecutan sin problema:

Spring @import

Sin embargo si tenemos decenas de beans ,esto acaba siendo problemático ya que es difícil clarificar que es todo lo que tenemos configurado. Para solventar este tipo de problemas Spring soporta la anotación @import . Con Spring @import podemos partir el fichero en varias partes y organizarlo todo mucho mejor.



Vamos a verlo en código:

```
package com.arquitecturajava.spring;

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

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SpringConfiguracionSeguridad {

	@Bean
	public static HelperSeguridad helper2() {

		return new HelperSeguridad();
	}

}
```

```
package com.arquitecturajava.spring;

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

@Configuration

public class SpringConfiguracionPersistencia {

	@Bean
	public static HelperPersistencia helper1() {

		return new HelperPersistencia();
	}
```

```
}
```

```
package com.arquitecturajava.spring;

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

@Configuration
@Import({SpringConfiguracionSeguridad.class,SpringConfiguracionPersist
encia.class})
public class SpringConfiguracionMega {

}
```

Acabamos de configurar Spring Framework apoyándonos en tres ficheros (SpringConfiguracionSeguridad, SpringConfiguracionPersistencia, SpringConfiguracionMega) . De esta forma es mucho más fácil acceder a cada bloque de configuración organizado en las categorías que nosotros necesitemos.

Otros artículos relacionados:

1. Spring Boot AOP y rendimiento
2. Introducción a Spring Data y JPA
3. Spring Singleton vs Prototype
4. @Import anotación