http://blog.csdn.net/baple/article/details/16864835

<context:annotation-config> 是用于激活那些已经在**spring**容器里注册过的bean（无论是通过xml的方式还是通过package sanning的方式）上面的注解，是一个注解处理工具。
<context:component-scan>除了具有<context:annotation-config>的功能之外，<context:component-scan>还可以在指定的package下扫描以及注册javabean 。

下面我们通过例子来详细查看他们的区别，
有三个class A,B,C,并且B,C的对象被注入到A中.

下面我们通过例子来详细查看他们的区别，
有三个class A,B,C,并且B,C的对象被注入到A中.

```
1. package com.xxx;
2. public class B {
3.   public B() {
4.     System.out.println("creating bean B: "
+ this);
5.   }
6. }
7.
8. package com.xxx;
9. public class C {
10.   public C() {
11.     System.out.println("creating bean C:
" + this);
12.   }
13. }
14.
15. package com.yyy;
16. import com.xxx.B;
17. import com.xxx.C;
18. public class A {
19.   private B bbb;
20.   private C ccc;
21.   public A() {
22.     System.out.println("creating bean A:
" + this);
```

```
23.    }
24.    public void setBbb(B bbb) {
25.      System.out.println("setting A.bbb
with " + bbb);
26.      this.bbb = bbb;
27.    }
28.    public void setCcc(C ccc) {
29.      System.out.println("setting A.ccc
with " + ccc);
30.      this.ccc = ccc;
31.    }
32. }
```

在applicationContext.xml中加入下面的配置：

```
<bean id="bBean"class="com.xxx.B"/>
<bean id="cBean"class="com.xxx.C"/>
<bean id="aBean"class="com.yyy.A">
  <property name="bbb" ref="bBean"/>
  <property name="ccc" ref="cBean"/>
</bean>
```

加载applicationContext.xml配置文件，将得到下面的结果:

```
creating bean B: com.xxx.B@c2ff5
creating bean C: com.xxx.C@1e8a1f6
creating bean A: com.yyy.A@1e152c5
setting A.bbb with com.xxx.B@c2ff5
setting A.ccc with com.xxx.C@1e8a1f6
```

OK, 这个结果没什么好说的，就是完全通过xml的方式，不过太过时了，下面通过注解的方式来简化我们的xml配置文件
首先，我们使用autowire的方式将对象bbb和ccc注入到A中：

```
1. package com.yyy;
2. import
org.springframework.beans.factory.annotation.
Autowired;
3. import com.xxx.B;
4. import com.xxx.C;
5. public class A {
```

```
6.    private B bbb;
7.    private C ccc;
8.    public A() {
9.      System.out.println("creating bean A: "
+ this);
10.   }
11.   @Autowired
12.   public void setBbb(B bbb) {
13.     System.out.println("setting A.bbb
with " + bbb);
14.     this.bbb = bbb;
15.   }
16.   @Autowired
17.   public void setCcc(C ccc) {
18.     System.out.println("setting A.ccc
with " + ccc);
19.     this.ccc = ccc;
20.   }
21. }
```

然后，我们就可以从applicationContext.xml中移除下面的配置

```
<property name="bbb" ref="bBean"/>
<property name="ccc" ref="cBean"/>
```

移除之后，我们的applicationContext.xml配置文件就简化为下面的样子了

```
<bean id="bBean"class="com.xxx.B"/>
<bean id="cBean"class="com.xxx.C"/>
<bean id="aBean"class="com.yyy.A"/>
```

当我们加载applicationContext.xml配置文件之后，将得到下面的结果：

```
creating bean B: com.xxx.B@5e5a50
creating bean C: com.xxx.C@54a328
creating bean A: com.yyy.A@a3d4cf
```

OK, 结果是错误的的，究竟是因为什么呢？为什么我们的属性没有被注入进去呢？是因为注解本身并不能够做任何事情，它们只是最基本的组成部分，我们需要能够处理这些注解的处理工具来处理这些注解

这就是`<context:annotation-config>` 所做的事情

我们将applicationContext.xml配置文件作如下修改：

```
<context:annotation-config />
<bean id="bBean"class="com.xxx.B"/>
<bean id="cBean"class="com.xxx.C"/>
<bean id="aBean"class="com.yyy.A"/>
```

当我们加载applicationContext.xml配置文件之后，将得到下面的结果：

```
creating bean B: com.xxx.B@15663a2
creating bean C: com.xxx.C@cd5f8b
creating bean A: com.yyy.A@157aa53
setting A.bbb with com.xxx.B@15663a2
setting A.ccc with com.xxx.C@cd5f8b
```

OK, 结果正确了。

下面演示<context:annotation-config> 跟 <context:component-scan>区别：
但是如果我们将代码作如下修改：

```
package com.xxx;
1. import
org.springframework.stereotype.Component;
2. @Component
3. public class B {
4.   public B() {
5.     System.out.println("creating bean B: "
+ this);
6.   }
7. }
8.
9. package com.xxx;
10. import
org.springframework.stereotype.Component;
11. @Component
12. public class C {
13.   public C() {
14.     System.out.println("creating bean C:
" + this);
15.   }
16. }
17.
18. package com.yyy;
```

```
19. import
org.springframework.beans.factory.annotation.
Autowired;
20. import
org.springframework.stereotype.Component;
21. import com.xxx.B;
22. import com.xxx.C;
23. @Component
24. public class A {
25.   private B bbb;
26.   private C ccc;
27.   public A() {
28.     System.out.println("creating bean A:
" + this);
29.   }
30.   @Autowired
31.   public void setBbb(B bbb) {
32.     System.out.println("setting A.bbb
with " + bbb);
33.     this.bbb = bbb;
34.   }
35.   @Autowired
36.   public void setCcc(C ccc) {
37.     System.out.println("setting A.ccc
with " + ccc);
38.     this.ccc = ccc;
39.   }
40. }
```

applicationContext.xml配置文件修改为：

```
<context:annotation-config />
```

当我们加载applicationContext.xml配置文件之后，却没有任何输出，这是为什么呢？
那是因为<context:annotation-config />仅能够在已经在已经注册过的bean上面起作用。
对于没有在spring容器中注册的bean，它并不能执行任何操作。
但是不用担心，<context:component-scan>除了具有<context:annotation-config />的功能之外，还具有自动将带有@component,@service,@Repository等注解的对象注册到spring容器中的功能。
我们将applicationContext.xml配置文件作如下修改：

```
<context:component-scan base-package="com.xxx"/>
```

当我们加载applicationContext.xml的时候，会得到下面的结果：

```
creating bean B: com.xxx.B@1be0f0a
creating bean C: com.xxx.C@80d1ff
```

这是什么原因呢?
是因为我们仅仅扫描了com.xxx包及其子包的类，而class A是在com.yyy包下，所以就扫描不到了
下面我们在applicationContext.xml中把com.yyy也加入进来：

```
<context:component-scan base-package="com.xxx"/>

<context:component-scan base-package="com.xxx,com.yyy"/>
```

然后加载applicationContext.xml就会得到下面的结果：

```
creating bean B: com.xxx.B@cd5f8b
creating bean C: com.xxx.C@15ac3c9
creating bean A: com.yyy.A@ec4a87
setting A.bbb with com.xxx.B@cd5f8b
setting A.ccc with com.xxx.C@15ac3c9
```

哇，结果正确啦！
回头看下我们的applicationContext.xml文件，已经简化为：

```
<context:component-scan base-package="com.xxx"/>
<context:component-scan base-package="com.xxx,com.yyy"/>
```

那如果我们在applicationContext.xml手动加上下面的配置，也就是说既在applicationContext.xml中手动的注册了A的实例对象，同时，通过component-scan去扫描并注册B,C的对象

```
<context:component-scan base-package="com.xxx"/>
<bean id="aBean"class="com.yyy.A"/>
```

结果仍是正确的:

```
creating bean B: com.xxx.B@157aa53
creating bean C: com.xxx.C@ec4a87
creating bean A: com.yyy.A@1d64c37
setting A.bbb with com.xxx.B@157aa53
```

```
setting A.ccc with com.xxx.C@ec4a87
```

虽然class A并不是通过扫描的方式注册到容器中的，但是`<context:component-scan>`所产生的的处理那些注解的处理器工具，会处理所有绑定到容器上面的bean，不管是通过xml手动注册的还是通过scanning扫描注册的。
那么，如果我们通过下面的方式呢？我们既配置了<context:annotation-config />，又配置了<context:component-scan base-package="com.xxx" />，它们都具有处理在容器中注册的bean里面的注解的功能。会不会出现重复注入的情况呢？

```
<context:annotation-config /><context:component-scan base-package="com.xxx"/><bean id="aBean"class="com.yyy.A"/>
```

不用担心，不会出现的：

```
creating bean B: com.xxx.B@157aa53
creating bean C: com.xxx.C@ec4a87
creating bean A: com.yyy.A@1d64c37
setting A.bbb with com.xxx.B@157aa53
setting A.ccc with com.xxx.C@ec4a87
```

因为`<context:annotation-config />`和`<context:component-scan>`同时存在的时候，前者会被忽略。也就是那些`@autowire`，`@resource`等注入注解只会被注入一次
哪怕是你手动的注册了多个处理器，spring仍然只会处理一次：

```
1. <context:annotation-config />
2. <context:component-scan base-package="com.xxx" />
3. <bean id="aBean" class="com.yyy.A" />
4. <bean id="bla" class="org.springframework.beans.factory
5. .annotation.AutowiredAnnotationBeanPostProcessor" />
6. <bean id="bla1" class="org.springframework.beans.factory
7. .annotation.AutowiredAnnotationBeanPostProcessor" />
8. <bean id="bla2" class="org.springframework.beans.factory
```

```
9.
.annotation.AutowiredAnnotationBeanPostProces
sor" />
10. <bean id="bla3"
class="org.springframework.beans.factory
11.
.annotation.AutowiredAnnotationBeanPostProces
sor" />
```

结果仍是正确的：

```
creating bean B: com.xxx.B@157aa53
creating bean C: com.xxx.C@ec4a87
creating bean A: com.yyy.A@25d2b2
setting A.bbb with com.xxx.B@157aa53
setting A.ccc with com.xxx.C@ec4a87
```