Spring Bean元素的相关属性depend-on和...

无依赖bean的创建和销毁顺序

我们都知道Spring的IOC容器提供的对bean的维护和管理功能，那么bean的创建顺序是如何呢？我们首先来看没有任何依赖关系的Bean的创建和销毁顺序。

我们以XML方式定义3个bean:

```
<?xml version="1.0" encoding="UTF-8"?>
<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"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http:
    <bean id="a" class="com.etoak.student.entity.A" destroy-method="des"></b
    <bean id="b" class="com.etoak.student.entity.B" destroy-method="des"></b
    <bean id="c" class="com.etoak.student.entity.C" destroy-method="des"></b
</beans>
```

> 注意：destroy-method是用来给每个bean指定销毁方法，即当容器调用close方法关闭时，spring会调用容器中所有bean的destroy方法，做一些销毁资源等清理的工作。

上面三个bean的定义顺序是：a、b、c,对应的java代码如下：

```
package com.etoak.student.entity;
public class A {
    public A(){
        System.out.println("A..contructed..");
    }
    public void des(){
        System.out.println("A...des。。");
    }
}
//~~~~~~~~~~~~~~~华丽的分割线~~~~~~~~~~~~~~~~~~~~~~~~
package com.etoak.student.entity;
public class B {
    public B(){
        System.out.println("B..contructed..");
    }
    public void des(){
        System.out.println("B...des。。");
    }
}
```

```
//~~~~~~~~~~~~~~~华丽的分割线~~~~~~~~~~~~~~~~~~~~~~~
package com.etoak.student.entity;
public class C {
    public C(){
        System.out.println("C..contructed..");
    }
    public void des(){
        System.out.println("C...des。。");
    }
}
```

在测试类中我们启动和关闭spring容器，分别查看bean的创建和关闭顺序。

```
package com.etoak.student.test;
import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {
    public static void main(String[] args) {
        System.out.println("1.准备构造ApplicationContext");
        AbstractApplicationContext ac = new ClassPathXmlApplicationContext('
        System.out.println("2.ApplicationContext已经构造完毕,接下来关闭容器");
        ac.close();
        System.out.println("3.容器已经关闭");
    }
}
```

控制台输出如下：

```
1.准备构造ApplicationContext
A..contructed..
B..contructed..
C..contructed..
2.ApplicationContext已经构造完毕,接下来关闭容器
C...des。。
B...des。。
A...des。。
3.容器已经关闭
```

通过以上测试我们非常容易发现【没有任何关联的bean】：

1. bean对象的创建顺序和applicationContex.xml中定义的顺序一致。
2. bean对象的销毁顺序和applicationContext.xml中定义的顺序相反。

使用depend-on影响bean的创建和销毁顺序

在无依赖关系的bean,创建顺序时通过xml的定义来影响的，如果不想调整XML中的bean的定义顺序或者无法调整bean的定义顺序，spring也提供了depend-on来声明某个bean在创建之前需要创建的bean，以及销毁时在当前bean销毁之后depend-on的bean再进行销毁。

```
<?xml version="1.0" encoding="UTF-8"?>
<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"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xsi:schemaLocation="http://www.springframework.org/schema/beans http:

    <bean id="a" class="com.etoak.student.entity.A" destroy-method="des" dep
    <bean id="b" class="com.etoak.student.entity.B" destroy-method="des"  de
    <bean id="c" class="com.etoak.student.entity.C" destroy-method="des" ></
    <bean id="d" class="com.etoak.student.entity.D" destroy-method="des"></b
    <bean id="e" class="com.etoak.student.entity.E" destroy-method="des"></b
```

运行测试输出：

```
Connected to the target VM, address: '127.0.0.1:53688', transport: 'socket'
1.准备构造ApplicationContext
D..contructed..
C..contructed..
E..contructed..
B..contructed..
A..contructed..
2.ApplicationContext已经构造完毕,接下来关闭容器
A...des。。
B...des。。
E...des。。
C...des。。
D...des。。
3.容器已经关闭
```

> depend-on：用来设置当前bean依赖的bean的名称，可以指定多个，多个之间使用逗号、分号、空格分隔开，如果依赖多个bean，则bean的顺序也是按照depend-on中的顺序依次构造。在上面例子中：beanA依赖beanB，beanB依赖d,c,e，所以构造时顺序D>C>E>B>A，销毁顺序：A>B>E>C>D.

补充：
在spring3.0之后，spring-context包提供了注解@DependsOn,其作用是在使用注解扫描bean时，指定bean依赖的对象。

```
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface DependsOn {

   String[] value() default {};

}
/*
可以用在类和方法上。
value：string类型的数组，用来指定当前bean需要依赖的bean名称，可以确保当前容器在创建
*/
```

用在类上

```java
package com.etoak.student.entity;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.DependsOn;
import org.springframework.stereotype.Component;

@DependsOn({"b","c"})
@Component
public class A {
    public A(){
        System.out.println("A..contructed..");
    }
    public void des(){
        System.out.println("A...des。。");
    }
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
package com.etoak.student.entity;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;
@Component("b")
public class B {
    public B(){
        System.out.println("B..contructed..");
    }
    public void des(){
        System.out.println("B...des。。");
    }
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
package com.etoak.student.entity;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;
@Component("c")
public class C {
    public C(){
        System.out.println("C..contructed..");
    }
    public void des(){
        System.out.println("C...des。。");
    }
}

```

用在方法上

```java
package com.etoak.student.entity;

```

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.DependsOn;
import org.springframework.stereotype.Component;

@DependsOn({"b","c"})
@Component
public class A {
    public A(){
        System.out.println("A..contructed..");
    }
    public void des(){
        System.out.println("A...des。。");
    }

    @Bean("d")
    @DependsOn("e")
    public D d(){
        D d = new D();
        return d;
    }
}
```

运行结果：

```
1.准备构造ApplicationContext
B..contructed..
C..contructed..
A..contructed..
E..contructed..
D..contructed..
2.ApplicationContext已经构造完毕,接下来关闭容器
3.容器已经关闭
```

总结：

不管是bean的属性depend-on，还是注解@DependsOn都是规定bean的构造和销毁顺序的。

@DependsOn的实现源码:


.11-sources.jar org springframework beans factory support AbstractBeanFactory
A.java ConfigurationClassPostProcessor.java AnnotatedBeanDefinition.java ConfigurationClassUtils.java ConfigurationClassEnhancer.java AbstractBeanFactory.java
Match Case Words Regex 5 matches
if (requiredType != null) {
beanCreation.tag("beanType", requiredType::toString);
}
RootBeanDefinition mbd = getMergedLocalBeanDefinition(beanName);
checkMergedBeanDefinition(mbd, beanName, args);
// Guarantee initialization of beans that the current bean depends on.
String[] dependsOn = mbd.getDependsOn();
获得依赖的bean数组
if (dependsOn != null) {
for (String dep : dependsOn) {
if (isDependent(beanName, dep)) {
如果循环依赖抛异常
throw new BeanCreationException(mbd.getResourceDescription()
"Circular depends-on relationship between '" + b
}
registerDependentBean(dep, beanName);
try {
getBean(dep);
递归调用getBean获取对象
这里是@DependOn的实现原理
}
catch (NoSuchBeanDefinitionException ex) {
throw new BeanCreationException(mbd.getResourceDescription()
"'" + beanName + "' depends on missing bean '" +
AbstractBeanFactory › doGetBean()