Java Web学习笔记

# 1 Spring学习的相关知识

## bean中id和name的区别

id必须符合xml规范，不能为特殊字符。在xml文件中唯一，每个bean的id只能有一个。

name可以包含特殊字符。在每个bean的name可以有多个，按照如下格式：<bean name="name1,name2" class="xxx.xxxxx"/>。

当其他bean需要引用某一个bean的时候，只能使用id。name只是一个别名（alias）。

## Spring框架初始化时xml文件的路径使用问题

Spring框架初始化的时，可以最好使用项目路径，而不是绝对路径，因为在不同的系统下，文件系统的格式是不一样的。

使用项目路径，需要使用ClassPathXmlApplicationContext类来创建BeanFactory对象。

项目路径的方式：BeanFactory factory = new ClassPathXmlApplicationContext(new String[]{"applicationContext.xml"});

## 使用工厂获取bean的实例的2种方法

### 方法一：工厂类中的获取实例方法为静态方法。

工厂类代码：

packag com.wzd;

public class BeanTestFactory{

public static BeanTest getBeanTestInstance(){

return new BeanTest();

}

}

spring配置文件：

<bean id="bean\_id" class="com.wzd.BeanTestFactory" factory-method="getBeanTestInstance"/>

### 方法二：工厂类中的获取实例方法为非静态方法。

工厂类代码：

packag com.wzd;

public class BeanTestFactory{

public BeanTest getBeanTestInstance(){

return new BeanTest();

}

}

spring配置文件：

<bean id="beanFactory\_id" class="com.wzd.BeanTestFactory"/> <!-- 工厂类的bean -->

<bean id="bean\_id" factory-bean="beanFactory\_id" factory-method="getBeanTestInstance"/> <!-- 使用工厂类的bean来创建实际对象的bean -->

## Spring中如何设置是否使用单例模式

在bean中设置scope属性可以设置是否使用单例模式，scope属性有四种可选值：prototype、singleton、request、session。

其中request、session在web开发中才会用到。

prototype -原型，表示每次获取bean实例都会创建一个新的对象。

singleton -单例，表示每次获取bean实例都是同一个，整个容器中该bean实例只有一个。

Spring中默认是使用单例模式，即singleton。

设置成非单例模式：

<bean id="daoImpl" class="wdz.pkg.daoImpl.DAOImpl" scope="prototype"></bean>

## bean什么时候被实例化？

#### 1.实例化时间测试

测试方法：在bean的构造方法中添加打印信息，通过debug单步调试查看。

1. bean为singleton模式，在容器实例化时bean被实例化。

注意：如果修改bean的属性lazy-init为true，则会在第一次获取该bean对象的时候进行实例化。

例如：

<bean id="serviceImpl" class="wzd.pkg.serviceImpl.ServiceImpl" lazy-init="true">

1. bean为prototype模式，在每次需要获取新的bean对象时，即每次调用BeanFactory对象的getBean()方法时会实例化一个新的对象。

#### 2. lazy-init属性的使用：

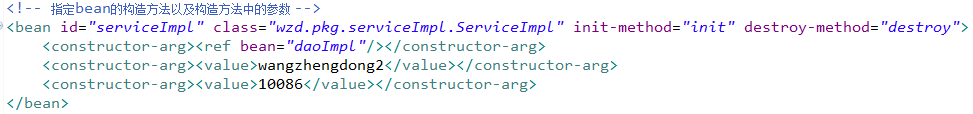
lazy-init不仅存在于bean标签中，也存在于beans标签中。在bean标签中，则表示该bean是需要根据lazy-nit属性值选择初始化时间，其他bean不受影响；在beans标签中，则表示该beans下所有的bean按照该lazy-init属性值来选择初始化时间。

建议不要将该属性设置为true。

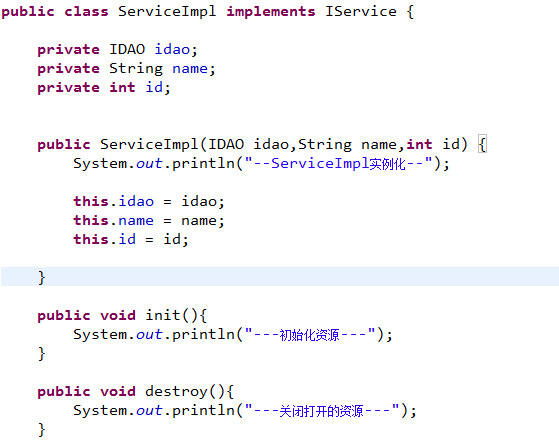
## 指定初始化方法和销毁方法

在bean中，init-method属性可以指定初始化方法，destroy-method属性可以指定销毁方法。

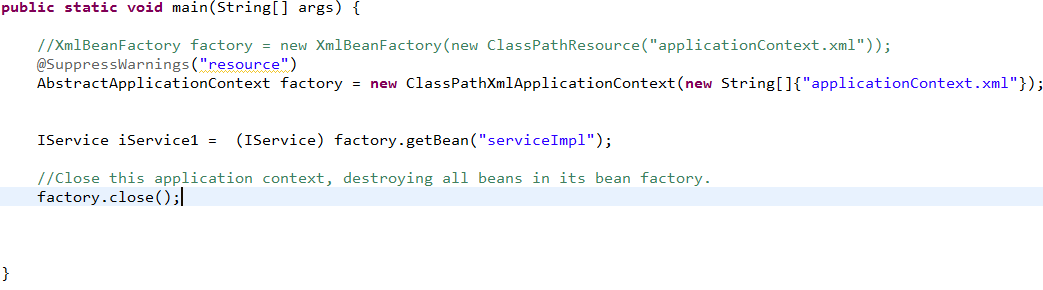
设置Spring的配置文件中的Bean如下：



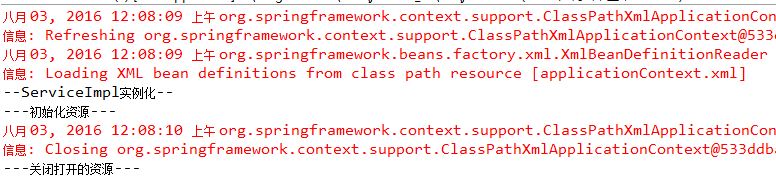
在wzd.pkg.serviceImpl.ServiceImp类中添加这2种方法，如下：



在测试类中需要调用applicationContext的正常关闭方法，代码如下：



运行结果如下：



结论：初始化方法在构造方法之后被调用，正常关闭Spring容器，bean的销毁方法将会在Spring容器关闭后被调用。

## bean属性的注入

在bean中会定义一些属性，那么如何指定这些属性的引用呢？有2种方法，第一，在实例化Bean对象的时候，通过构造方法传入对象的实例；第二，通过该属性的setter方法对其进行间接赋值，实际上就是在Spring的xml配置文件中使用<bean/>标签的子标签<property/>进行设置。

#### 通过构造方法进行注入<constructor-arg>

Bean的定义如下，其中有参构造方法中设置了类中的三个属性：



三个属性中，变量idao需要引用IDAO的实例；name可以直接赋值，也可以指向String实例；id是元类型int，只能直接赋值。引用使用<constructor-arg/>标签中的ref属性（或子标签），直接赋值使用< constructor-arg />标签中的value属性（或子标签）。

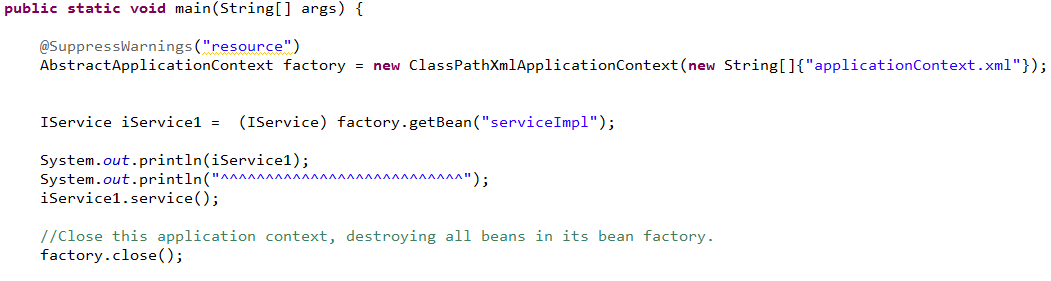
Spring的配置文件内容如下：



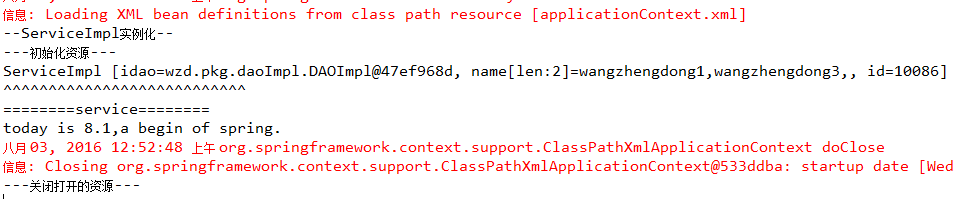
使用<bean/>标签的子标签<constructor-arg/>进行注入构造实参。当有多个参数的时，使用多个<constructor-arg/>标签。

注意：多个<constructor-arg/>的顺序应该和构造方法中参数的顺序相同。

实例化Spring容器，并获取id为serviceImpl的bean，代码如下：



运行得到的结果如下：



#### 通过<property/>标签进行注入<property/>

注入属性的时候，使用的是<property/>标签，引用使用<property/>标签中的ref属性（或子标签），直接赋值使用<property/>标签中的value属性（或子标签）。

Bean的代码和上面的相同，Spring的配置文件中的内容不同，如下：



实例化Spring容器，并获取id为serviceImpl的bean，代码和上面相同。

运行结果为：

