08--BeanFactory和FactoryBean的区别

2018年11月01日 15:24:24 闲来也无事 阅读数 3000 更多

BeanFactory和FactoryBean是两个容易混淆的概念，很多人喜欢问两者之间的区别，其实两者之间并无内在联系。

- BeanFactory接口：IoC容器的顶级接口，是IoC容器的最基础实现，也是访问Spring容器的根接口，负责对bean的创建，访问等工作。
- FactoryBean接口：可以返回bean的实例的工厂bean，通过实现该接口可以对bean进行一些额外的操作，例如根据不同的配置类型返回不同类型的be
  配置等。在使用上也有些特殊，BeanFactory接口中有一个字符常量 `String FACTORY_BEAN_PREFIX = "&";` 当我们去获取BeanFactory类型的bean时
  beanName不加&则获取到对应bean的实例；如果beanName加上&，则获取到BeanFactory本身的实例；FactoryBean接口对应Spring框架来说占有重
  Spring本身就提供了70多个FactoryBean的实现。他们隐藏了实例化一些复杂的细节，给上层应用带来了便利。从Spring3.0开始，FactoryBean开始支

下面来看FactoryBean的使用方式

1.简化xml配置，隐藏细节

使用Setter方法注入大量属性会造成配置文件臃肿，这时可以考虑使用FactoryBean来简化配置。

- bean

```
1  package com.lyc.cn.v2.day01.factoryBean;
2
3  /**
4   * @author: LiYanChao
5   * @create: 2018-09-05 11:50
6   */
7  public class Student {
8      /** 姓名 */
9      private String name;
10     /** 年龄 */
11     private int age;
12     /** 班级名称 */
13     private String className;
14
15     // ...可能能会有更多的属性
16
17     public Student() {
18     }
19
20     public Student(String name, int age, String className) {
21         this.name = name;
22         this.age = age;
23         this.className = className;
24     }
25
26     public String getName() {
27         return name;
28     }
29
30     public void setName(String name) {
31         this.name = name;
32     }
33
34     public int getAge() {
35         return age;
36     }
37
38     public void setAge(int age) {
39         this.age = age;
40     }
41
42     public String getClassName() {
```

```
        return className;
    }

    public void setClassName(String className) {
        this.className = className;
    }

    @Override
    public String toString() {
        return "Student{" + "name='" + name + '\'' + ", age=" + age + ", className='" + className + '\'' + '}';
    }
}

```

```
package com.lyc.cn.v2.day01.factoryBean;

import org.springframework.beans.factory.FactoryBean;

/**
 * @author: LiYanChao
 * @create: 2018-09-05 11:49
 */
public class StudentFactoryBean implements FactoryBean<Student> {

    private String studentInfo;

    @Override
    public Student getObject() throws Exception {
        if (this.studentInfo == null) {
            throw new IllegalArgumentException("'studentInfo' is required");
        }

        // 分割属性
        String[] splitStudentInfo = studentInfo.split(",");
        if (null == splitStudentInfo || splitStudentInfo.length != 3) {
            throw new IllegalArgumentException("'studentInfo' config error");
        }

        // 创建Student并填充属性
        Student student = new Student();
        student.setName(splitStudentInfo[0]);
        student.setAge(Integer.valueOf(splitStudentInfo[1]));
        student.setClassName(splitStudentInfo[2]);
        return student;
    }

    @Override
    public Class<?> getObjectType() {
        return StudentFactoryBean.class;
    }

    @Override
    public boolean isSingleton() {
        return true;
    }

    public void setStudentInfo(String studentInfo) {
        this.studentInfo = studentInfo;
    }
}

```

Student是一个普通的类，StudentFactoryBean实现了FactoryBean接口，是一个FactoryBean。

- xml

```
<bean id="student" class="com.lyc.cn.v2.day01.factoryBean.StudentFactoryBean" p:studentInfo="张三,25,三年二班"/>
```

- 测试

```
//FactoryBean简化配置测试
System.out.println(xmlBeanFactory.getBean("student"));
System.out.println(xmlBeanFactory.getBean("&student"));
```

- 结果

```
========测试方法开始=======

Student{name='张三', age=25, className='三年二班'}
com.lyc.cn.v2.day01.factoryBean.StudentFactoryBean@1ff8b8f

========测试方法结束=======
```

- 分析
  `xmlBeanFactory.getBean("student")` 获取到的是StudentFactoryBean产生的实例，也就是Student类的实例；而 `xmlBeanFactory.getBean("&st` 到的是StudentFactoryBean自己的实例。

2.返回不同Bean的实例

- bean

```
package com.lyc.cn.v2.day01.factoryBean;

/**
 * 家具接口
 */
public interface Furniture {
    void sayHello();
}

```

```
package com.lyc.cn.v2.day01.factoryBean;

/**
 * 椅子
 * @author: LiYanChao
 * @create: 2018-09-30 17:35
 */
public class Chair implements Furniture{

    @Override
    public void sayHello() {
        System.out.println("我是一把椅子。");
    }
}

```

```
package com.lyc.cn.v2.day01.factoryBean;

/**
 * 桌子
 * @author: LiYanChao
 * @create: 2018-09-30 17:35
 */
public class Desk implements Furniture{

    @Override
    public void sayHello() {
        System.out.println("我是一个桌子。");
    }
}

```

```
package com.lyc.cn.v2.day01.factoryBean;

import org.springframework.beans.factory.FactoryBean;
```

```
4
5  /**
6   * 家具工厂bean
7   * @author: LiYanChao
8   * @create: 2018-09-05 15:11
9   */
10 public class FurnitureFactoryBean implements FactoryBean<Furniture> {
11
12     private String furniture;
13
14     @Override
15     public Furniture getObject() throws Exception {
16         if (null == furniture) {
17             throw new IllegalArgumentException("'furniture' is required");
18         }
19         if ("chair".equals(furniture)) {
20             return new Chair();
21         } else if ("desk".equals(furniture)) {
22             return new Desk();
23         } else {
24             throw new IllegalArgumentException("'furniture' type error");
25         }
26     }
27
28     @Override
29     public Class<?> getObjectType() {
30         if (null == furniture) {
31             throw new IllegalArgumentException("'furniture' is required");
32         }
33         if ("chair".equals(furniture)) {
34             return Chair.class;
35         } else if ("desk".equals(furniture)) {
36             return Desk.class;
37         } else {
38             throw new IllegalArgumentException("'furniture' type error");
39         }
40     }
41
42     @Override
43     public boolean isSingleton() {
44         return true;
45     }
46
47     public void setFurniture(String furniture) {
48         this.furniture = furniture;
49     }
50 }
```

- xml

```
1 <bean id="furniture" class="com.lyc.cn.v2.day01.factoryBean.FurnitureFactoryBean" p:furniture="desk"/>
```

- 测试

```
1 @Test
2 public void test12() {
3     //FactoryBean简单工厂测试
4     Furniture furniture = xmlBeanFactory.getBean("furniture", Furniture.class);
5     furniture.sayHello();
6 }
```

- 结果

```
1 ========测试方法开始=======
2
3 我是一个桌子。
4
5 ========测试方法结束=======
```

- 说明
  新建了家具接接口和桌子、椅子实现类，通过xml文件配置，在FurnitureFactoryBean的getObject方法进行判断，并返回不同的家具类型实例。

3.FactoryBean源码

该接口的源码比较少，只声明了三个接口

```
public interface FactoryBean<T> {

    //返回此工厂管理的对象的实例（可能Singleton和Prototype）
    //如果此FactoryBean在调用时尚未完全初始化（例如，因为它涉及循环引用），则抛出相应的FactoryBeanNotInitializedException。
    //从Spring 2.0开始，允许FactoryBeans返回null 对象。工厂会将此视为正常使用价值; 在这种情况下，它不再抛出FactoryBeanNotInitializedE
    //鼓励FactoryBean实现现在自己抛出FactoryBeanNotInitializedException，视情况而定。
    T getObject() throws Exception;

    //返回此FactoryBean创建的对象类型,默认返回null
    Class<?> getObjectType();

    //实例是否单例模式,默认返回true
    default boolean isSingleton() {
        return true;
    }

}
```