###### 默认装配方式

代码通过getBean()方式从容器获取指定的Bean实例，容器首先会调用Bean类的无参构造器，创建空值的实例对象。

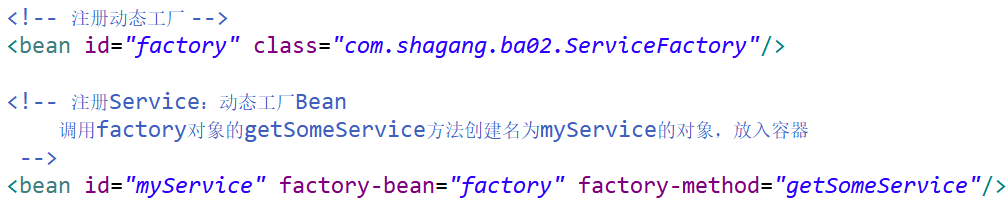
###### 动态工厂Bean

有些时候，项目中需要通过工厂类来创建Bean实例，而不能像前面例子中似的，直接

由Spring容器来装配Bean实例。使用工厂模式创建Bean实例，就会使工厂类与要创建的Bean类耦合到一起。

1. **将动态工厂Bean作为普通Bean使用**
2. **使用Spring的动态工厂Bean**

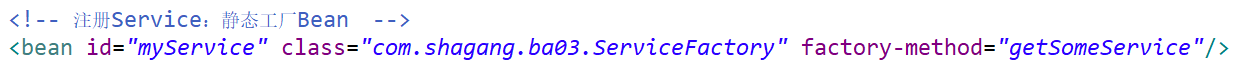
Spring对于使用动态工厂来创建的Bean，有专门的属性定义。factory-bean指定相应的工厂Bean，由factory-method指定创建所用方法。此时配置文件中至少会有两个Bean的定义：工厂类的Bean，与工厂类所要创建的目标类Bean。而测试类中不再需要获取工厂Bean对象了，可以直接获取目标Bean对象。实现测试类与工厂类间的解耦。



###### 静态工厂Bean

静态工厂无需工厂实例，所以不再需要定义静态工厂<bean/>。

而对于工厂所要创建的Bean，其并不是由自己的类创建的，所以无需指定自己的类。但是其是由工厂类创建的，所以需要工厂指定所用工厂类。故class属性指定的是工厂类而非自己的类。当然，还需要通过factory-method属性指定工厂方法。



###### 容器中Bean的作用域

当通过Spring容器创建一个Bean实例时，不经可以完成Bean的实例化，还可以通过scope属性，为Bean指定特定的作用域。Spring支持五种作用域。

1. singleton：单态模式。即整个Spring容器中，使用singleton定义的Bean将是单例的，新的实例。
2. prototype：原型模式。即每次使用getBean方法获取同一个<bean/>的实例都是一个新的实例。
3. request：对于每次HTTP请求，都将会产生一个不同的Bean实例。
4. session：对于每个不同的HTTP session，都将会产生一个不同的Bean实例。

注意：

* 对于scope的值request、session与global session，只有在Web应用中使用Spring时，该作用域才有效。
* 对于scope为singleton的单例模式，该Bean是在被容器创建时即被装配好了。
* 对于scope为prototype的原型模式，Bean实例是在代码中使用该Bean实例时才进行装配的。

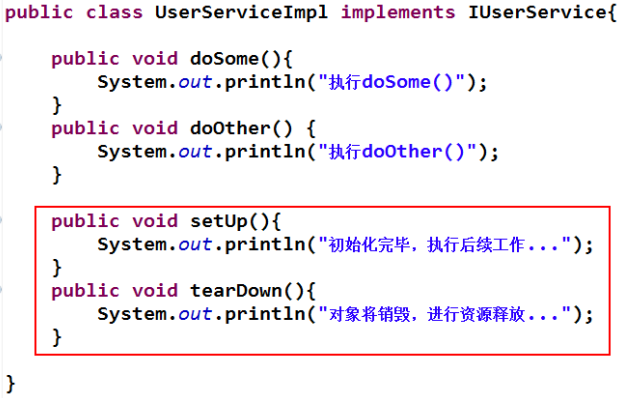
###### Bean后处理器

Bean后处理器是一个Bean，容器中所有在初始化时，均会自动执行该类的两个方法。由于该Bean是由其他Bean自动调用执行的，不是程序员手工调用，故此B二胺、无需id属性。

###### 定制Bean的生命始末

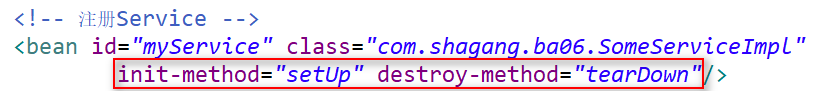
可以为Bean定制初始化后的行为，也可以为Bean定制销毁前的生命行为。

需要现在Bean类中事先定义好：是方法名随意的public viod方法。



其次，在配置文件的<bean/>标签中增加如下属性：

* init-method：指定初始化方法的方法名；
* destroy-method：指定销毁方法的方法名。



注：若要看到Bean的destroy-method的执行结果，需满足两个条件：

1. Bean为singleton，即单例
2. 要确保容器关闭。接口ApplicationContext没有close()方法，但其实现类有。所以，可以将ApplicationContext强转为其实现类对象ClassPathXmlApplicationContext，或直接创建的就是实现类对象。

###### Bean的生命周期



###### <bean/>标签的id属性与name属性

一般情况下，命名<bean/>使用id属性，而不使用name属性。在没有id属性的情况下，name属性与id属性作用是相同的。但，当<bean/>中包含有一些特殊字符时，就需要使用name属性了。

id的命名需要满足XML对ID属性命名规范：必须以字母开头，可以包含字母、数字、下划线、连字符、句号、冒号。

name属性值则可以包含各种字符。