BeanFactory是spring中比较原始的Factory。如XMLBeanFactory就是一种典型的BeanFactory。原始的BeanFactory无法支持spring的许多插件，如AOP功能、Web应用等。

ApplicationContext接口，它由BeanFactory接口派生而来。ApplicationContext包含BeanFactory的所有功能，通常建议比BeanFactory优先。

### 1、BeanFactory和FactoryBean的区别

* BeanFactory是接口，提供了IOC容器最基本的形式，给具体的IOC容器的实现提供了规范；
* FactoryBean也是接口，为IOC容器中Bean的实现提供了更加灵活的方式，FactoryBean在IOC容器的基础上给Bean的实现加上了一个简单工厂模式和装饰模式(如果想了解装饰模式参考：修饰者模式(装饰者模式，Decoration) 我们可以在getObject()方法中灵活配置。其实在Spring源码中有很多FactoryBean的实现类.

区别：

* BeanFactory是个Factory，也就是IOC容器或对象工厂，FactoryBean是个Bean。在Spring中，所有的Bean都是由BeanFactory(也就是IOC容器)来进行管理的。
* 对FactoryBean而言，这个Bean不是简单的Bean，而是一个能生产或者修饰对象生成的工厂Bean，它的实现与设计模式中的工厂模式和修饰器模式类似。

### 1、BeanFactory

BeanFactory，以Factory结尾，表示它是一个工厂类(接口)， 它负责生产和管理bean的一个工厂。在Spring中，BeanFactory是IOC容器的核心接口，它的职责包括：实例化、定位、配置应用程序中的对象及建立这些对象间的依赖。

BeanFactory只是个接口，并不是IOC容器的具体实现，但是Spring容器给出了很多种实现，如 DefaultListableBeanFactory、XmlBeanFactory、ApplicationContext等，其中XmlBeanFactory就是常用的一个，该实现将以XML方式描述组成应用的对象及对象间的依赖关系。XmlBeanFactory类将持有此XML配置元数据，并用它来构建一个完全可配置的系统或应用。

都是附加了某种功能的实现。它为其他具体的IOC容器提供了最基本的规范，例如XmlBeanFactory，ApplicationContext 等具体的容器都是实现了BeanFactory，再在其基础之上附加了其他的功能。

BeanFactory和ApplicationContext就是spring框架的两个IOC容器，现在一般使用ApplicationContext，其不但包含了BeanFactory的作用，同时还进行更多的扩展。

原始的BeanFactory无法支持spring的许多插件，如AOP功能、Web应用等。ApplicationContext接口由BeanFactory接口派生而来，包含BeanFactory的所有功能，通常建议比BeanFactory优先。

ApplicationContext以一种更向面向框架的方式工作以及对上下文进行分层和实现继承，ApplicationContext包还提供了以下的功能：

<https://blog.csdn.net/qq_24313635/article/details/82112565>

* MessageSource，提供国际化的消息访问；
* 资源访问，如URL和文件；
* 事件传播；
* 载入多个（有继承关系）上下文，使得每一个上下文都专注于一个特定的层次，比如应用的web层。

使用getBean(String beanName)方法就可以取得bean的实例；BeanFactory提供的方法及其简单，仅提供了六种方法供客户调用：

* boolean containsBean(String beanName) 判断工厂中是否包含给定名称的bean定义，若有则返回true
* Object getBean(String) 返回给定名称注册的bean实例。根据bean的配置情况，如果是singleton模式将返回一个共享实例，否则将返回一个新建的实例，如果没有找到指定bean,该方法可能会抛出异常
* Object getBean(String, Class) 返回以给定名称注册的bean实例，并转换为给定class类型
* Class getType(String name) 返回给定名称的bean的Class,如果没有找到指定的bean实例，则排除NoSuchBeanDefinitionException异常
* boolean isSingleton(String) 判断给定名称的bean定义是否为单例模式
* String[] getAliases(String name) 返回给定bean名称的所有别名

### 2、FactoryBean

* TgetObject()：返回由FactoryBean创建的Bean实例，如果isSingleton()返回true，则该实例会放到Spring容器中单实例缓存池中；
* booleanisSingleton()：返回由FactoryBean创建的Bean实例的作用域是singleton还是prototype；
* Class<T>getObjectType()：返回FactoryBean创建的Bean类型。

一般情况下，Spring通过反射机制利用<bean>的class属性指定实现类实例化Bean，在某些情况下，实例化Bean过程比较复杂，如果按照传统的方式，则需要在<bean>中提供大量的配置信息。配置方式的灵活性是受限的，这时采用编码的方式可能会得到一个简单的方案。

Spring为此提供了一个org.springframework.bean.factory.FactoryBean的工厂类接口，用户可以通过实现该接口定制实例化Bean的逻辑。FactoryBean接口对于Spring框架来说占用重要的地位，Spring自身就提供了90多个FactoryBean的实现。它们隐藏了实例化一些复杂Bean的细节，给上层应用带来了便利。从Spring3.0开始，FactoryBean开始支持泛型，即接口声明改为FactoryBean<T>的形式。以Bean结尾，表示它是一个Bean，不同于普通Bean的是：它是实现了FactoryBean<T>接口的Bean，根据该Bean的ID从BeanFactory中获取的实际上是FactoryBean的getObject()返回的对象，而不是FactoryBean本身，如果要获取FactoryBean对象，请在id前面加一个&符号来获取。