spring IOC容器详解

# IOC结构体系

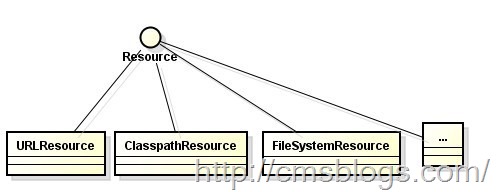
IOC作为一个容器，它里面放得都是bean、bean与bean之间的对应关系，而bean之间的对应关系我们开始都是通过xml配置文件来体现的。那么这里就反馈了如下几个问题：

1. 对应与对象之间的关系是通过xml配置文件来描述的（当然也可以是properties等文件）。
2. 描述的文件存放位置在那里，一般来说我们都是放在classpath目录下的，但是也可是是URL、fileSystem。
3. 文件的解析。
4. Bean在容器中的表现形式，也就是它的数据结构。

对于spring而言，它用Resource, BeanDefinition, BeanDefinitionReader, BeanFactory、ApplicationContext五个组件来实现以上问题，而同时这5个接口定义了 spring ioc 容器的基本代码组件结构。

## Resource

Resource，对资源的抽象，它的每一个实现类都代表了一种资源的访问策略，如ClasspathResource 、 URLResource ，FileSystemResource 等。

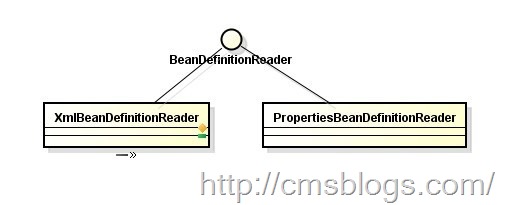


## BeanDefinition

用来描述和抽象一个具体的Bean对象，它是描述Bean对象的基本数据结构。

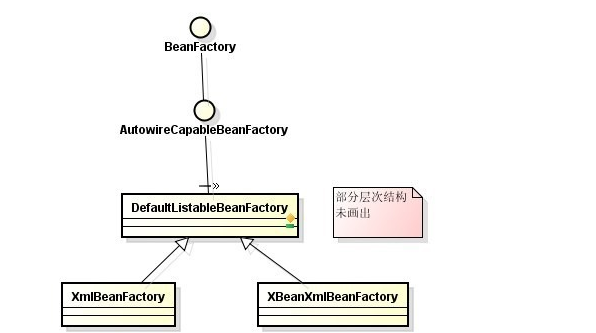
## BeanDefinitionReader

外部资源所表达的语义需要统一转化为统一的内部数据结构BeanDefinition，这个时候BeanDefinitionReader就起到统一解析的作用力了。对应不同的描述需要有不同的 Reader 。如 XmlBeanDefinitionReader 用来读取xml 描述配置的 bean 对象。



## BeanFactory

BeanFactory是一个非常纯粹的bean容器，它是IOC必备的数据结构，其中BeanDefinition是它的基本结构，它内部维护着一个BeanDefinition map，并可根据BeanDefinition 的描述进行 bean 的创建和管理。

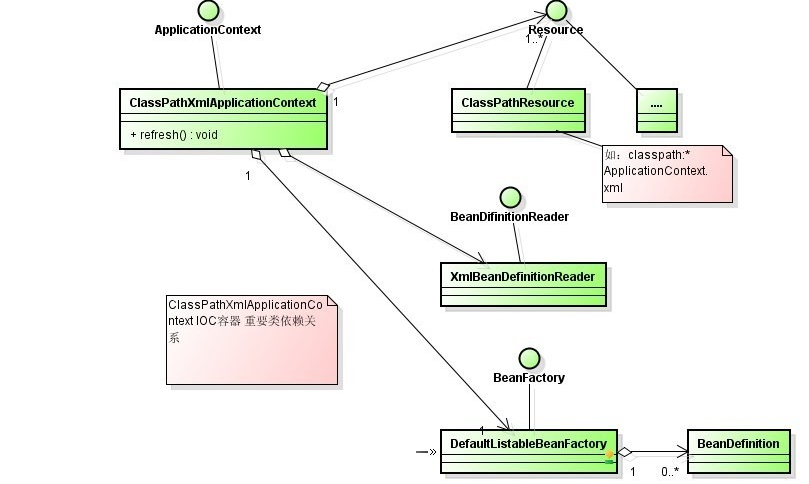


## ApplicationContext

这个就是大名鼎鼎的Spring容器，它叫做应用上下文，与我们应用息息相关，她继承BeanFactory，所以它是BeanFactory的扩展升级版，如果BeanFactory是屌丝的话，那么ApplicationContext则是名副其实的高富帅。由于ApplicationContext的结构就决定了它与BeanFactory的不同，其主要区别有：

1. 继承MessageSource，提供国际化的标准访问策略。
2. 继承ApplicationEventPublisher，提供强大的事件机制。
3. 扩展ResourceLoader，可以用来加载多个Resource，可以灵活访问不同的资源。
4. 对Web应用的支持。

下图是上面组合关系图（以ClasspathXmlApplicationContext 为例）



# spring ioc初始化过程

首先我们先来看看如下一段代码

|  |
| --- |
| ClassPathResource resource = new ClassPathResource("bean.xml");  DefaultListableBeanFactory factory = new DefaultListableBeanFactory();  XmlBeanDefinitionReader reader = new XmlBeanDefinitionReader(factory);  reader.loadBeanDefinitions(resource);  这段代码是编程式使用IOC容器，通过这个简单的程序我们初步判定IOC容器的使用过程：  第一步：创建IOC配置文件的资源，也就是Resource接口。  第二步：创建BeanFactory，DefaultListtableBeanFactory是BeanFactory模式实现类。  第三步：创建一个BeanDefinitionReader对象，该对象为BeanDefinition的读取器。xml文件就使用XMLBeanDefinitionReader。  第四步：使用Reader来装载配置文件。loadBeanDefinitions就包括了资源文件的解析和注入过程。  通过上面四个步骤我们就可以轻松地使用IOC容器了，在整个过程可以剖析为三个步骤，：Resource定位、载入、注册。如下： |

## Resource定位

我们一般使用外部资源来描述Bean对象，所以IOC容器第一步就是需要定位Resource外部资源。Resource的定位其实就是BeanDefinition的资源定位，它是由ResourceLoader通过统一的Resource接口来完成的，这个Resource对各种形式的BeanDefinition的使用都提供了统一接口。

## 载入

第二个过程就是BeanDefinition的载入。BeanDefinitionReader读取、解析Resource定位的资源，也就是将用户定义好的Bean表示成IOC容器的内部数据结构也就是BeanDefinition。在IOC容器内部维护着一个BeanDefinition Map的数据结构，通过这样的数据结构，IOC容器能够对Bean进行更好的管理。

在配置文件中每一个<bean>都对应着一个BeanDefinition对象。

## 注册

第三个过程则是注册，即向IOC容器注册这些BeanDefinition，这个过程是通过BeanDefinitionRegistery接口来实现的。在IOC容器内部其实是将第二个过程解析得到的BeanDefinition注入到一个HashMap容器中，IOC容器就是通过这个HashMap来维护这些BeanDefinition的。在这里需要注意的一点是这个过程并没有完成依赖注入，依赖注册是发生在应用第一次调用getBean向容器所要Bean时。当然我们可以通过设置预处理，即对某个Bean设置lazyinit属性，那么这个Bean的依赖注入就会在容器初始化的时候完成。

