**我们说 BeanFactory 是 Bean 容器，那么 Bean 又是什么呢？**

Bean 在代码层面上可以认为是 BeanDefinition 的实例, 我们自己定义的各个 Bean 其实会转换成一个个 BeanDefinition实例存在于BeanFactory中。

BeanDefinition 中保存了我们的Bean定义信息：

|  |
| --- |
| **public** **interface** BeanDefinition **extends** AttributeAccessor, BeanMetadataElement {    // Modifiable attributes  // 设置父 Bean，这里涉及到 bean 继承（继承父 Bean 的配置信息），不是 java 继承。  **void** setParentName(String parentName);  String getParentName();  // 用于反射生成实例  **void** setBeanClassName(String beanClassName);  String getBeanClassName();  // 默认只提供 sington 和 prototype 两种scope， 很多读者可能知道还有 request, session, globalSession, application, websocket 这几种，它们属于基于 web 的扩展。  **void** setScope(String scope);  String getScope();  **void** setLazyInit(**boolean** lazyInit);  **boolean** isLazyInit();  // 设置该 Bean 依赖的所有的 Bean  **void** setDependsOn(String... dependsOn);  String[] getDependsOn();  // 设置该 Bean 是否可以注入到其他 Bean 中，只对按类型注入有效， 如果按名称注入，即使这边设置了 false，也是可以的  **void** setAutowireCandidate(**boolean** autowireCandidate);  **boolean** isAutowireCandidate();  // 同一接口的多个实现，如果不指定名字的话，Spring 会优先选择设置 primary 为 true 的 bean  **void** setPrimary(**boolean** primary);  **boolean** isPrimary();  // 有些实例不是用反射生成的，而是用工厂模式生成的  **void** setFactoryBeanName(String factoryBeanName);  String getFactoryBeanName();  **void** setFactoryMethodName(String factoryMethodName);  String getFactoryMethodName();  // 构造器参数  ConstructorArgumentValues getConstructorArgumentValues();  // Bean 中的属性值，后面给 bean 注入属性值的时候会说到  MutablePropertyValues getPropertyValues(); |
| // Read-only attributes  **boolean** isSingleton();  **boolean** isPrototype();  // 如果这个 Bean 是被设置为 abstract，那么不能实例化，常用于作为 父bean 用于继承，很少用......  **boolean** isAbstract();  **int** getRole();  String getDescription();  String getResourceDescription();  BeanDefinition getOriginatingBeanDefinition();  } |