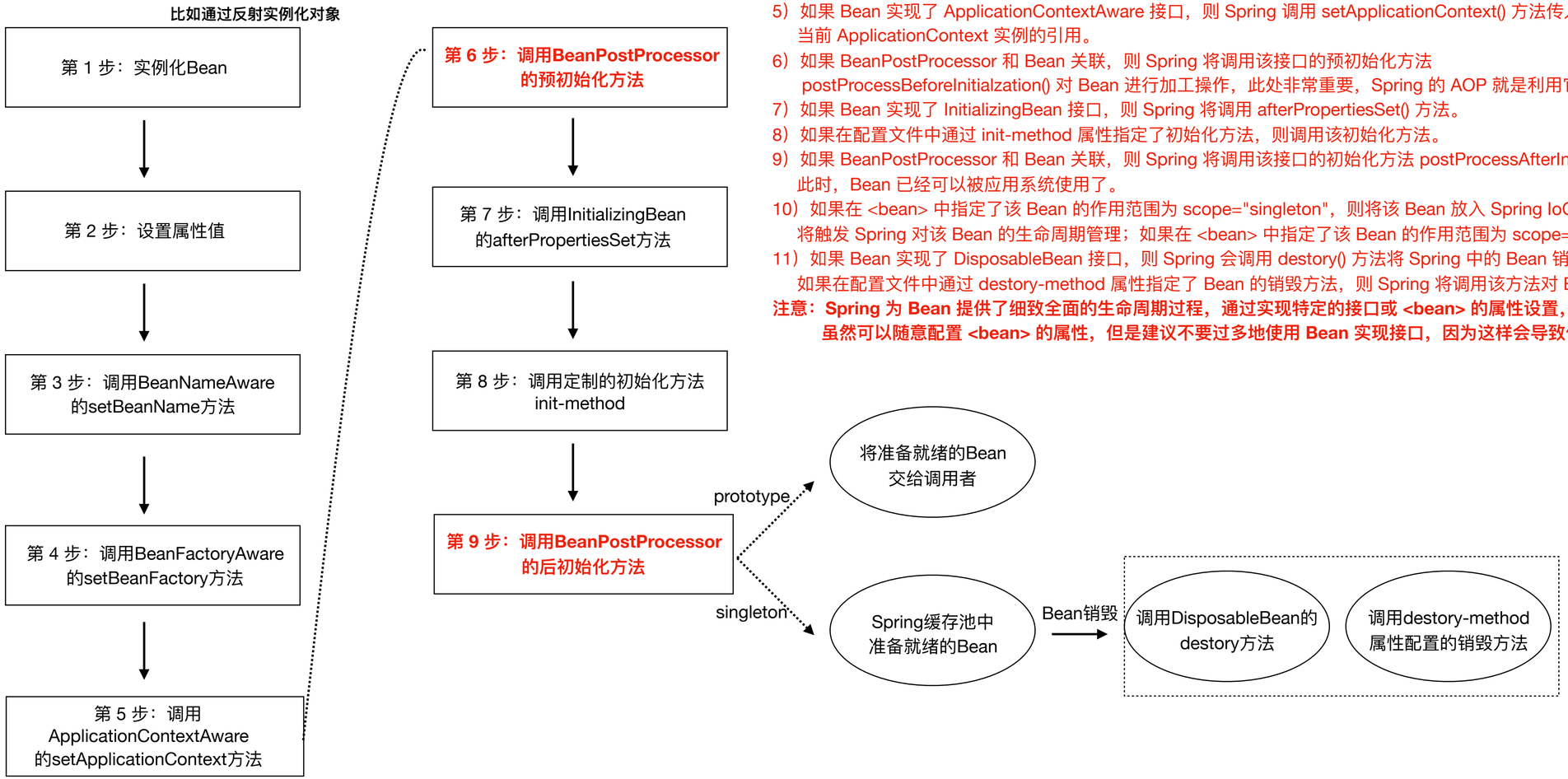
SpringBean的生命周期图

by 应癫



Bean 生命周期的整个执行过程描述：

1）根据配置情况调用 Bean 构造方法或工厂方法实例化 Bean。
2）利用依赖注入完成 Bean 中所有属性值的配置注入。
3）如果 Bean 实现了 BeanNameAware 接口，则 Spring 调用 Bean 的 setBeanName() 方法传入当前 Bean 的 id 值。
4）如果 Bean 实现了 BeanFactoryAware 接口，则 Spring 调用 setBeanFactory() 方法传入当前工厂实例的引用。
5）如果 Bean 实现了 ApplicationContextAware 接口，则 Spring 调用 setApplicationContext() 方法传入当前 ApplicationContext 实例的引用。
6）如果 BeanPostProcessor 和 Bean 关联，则 Spring 将调用该接口的预初始化方法 postProcessBeforeInitialzation() 对 Bean 进行加工操作，此处非常重要，Spring 的 AOP 就是利用它实现的。
7）如果 Bean 实现了 InitializingBean 接口，则 Spring 将调用 afterPropertiesSet() 方法。
8）如果在配置文件中通过 init-method 属性指定了初始化方法，则调用该初始化方法。
9）如果 BeanPostProcessor 和 Bean 关联，则 Spring 将调用该接口的初始化方法 postProcessAfterInitialization()。此时，Bean 已经可以被应用系统使用了。
10）如果在 <bean> 中指定了该 Bean 的作用范围为 scope="singleton"，则将该 Bean 放入 Spring IoC 的缓存池中，将触发 Spring 对该 Bean 的生命周期管理；如果在 <bean> 中指定了该 Bean 的作用范围为 scope="prototype"，则将该 Bean 交给调用者
11）如果 Bean 实现了 DisposableBean 接口，则 Spring 会调用 destory() 方法将 Spring 中的 Bean 销毁；如果在配置文件中通过 destory-method 属性指定了 Bean 的销毁方法，则 Spring 将调用该方法对 Bean 进行销毁。

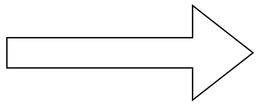
注意：Spring 为 Bean 提供了细致全面的生命周期过程，通过实现特定的接口或 <bean> 的属性设置，都可以对 Bean 的生命周期过程产生
虽然可以随意配置 <bean> 的属性，但是建议不要过多地使用 Bean 实现接口，因为这样会导致代码和 Spring 的聚合过于紧密

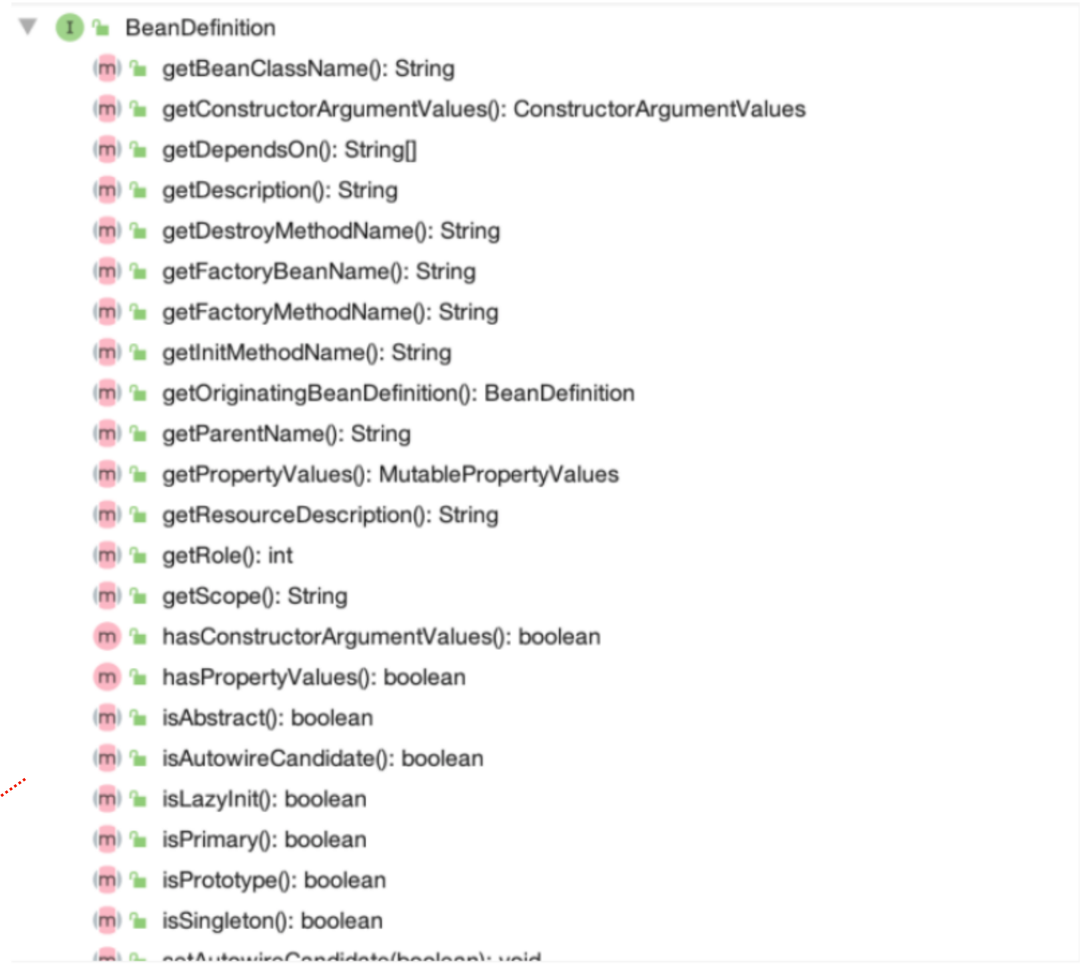
Spring容器启动的过程中，会将Bean解析成Spring内部的**BeanDefinition**结构

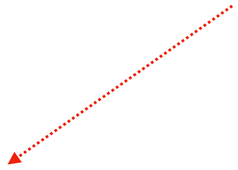
```xml
<bean id="transferService" class="com.lagou.edu.service.impl.TransferServiceImpl">
    <!--set+ name 之后锁定到传值的set方法了，通过反射技术可以调用该方法传入对应的值-->
    <property name="AccountDao" ref="accountDao"></property>
</bean>
```

```xml
<!--事务管理器-->
<bean id="transactionManager" class="com.lagou.edu.utils.TransactionManager">
    <property name="ConnectionUtils" ref="connectionUtils"/>
</bean>

<!--代理对象工厂-->
<bean id="proxyFactory" class="com.lagou.edu.factory.ProxyFactory">
    <property name="TransactionManager" ref="transactionManager"/>
</bean>
```







类名、scope、属性、构造函数参数列表、依赖的bean、是否是单例类、是否是懒加载等，
其实就是**将Bean的定义信息存储到这个BeanDefinition相应的属性中，后面对Bean的操作就直接对BeanDefinition进行，**
例如拿到这个BeanDefinition后，可以**根据里面的类名、构造函数、构造函数参数，使用反射进行对象创建。**