1.Spirng的InitializingBean为bean提供了定义初始化方法的方式。InitializingBean是一个接口，它仅仅包含一个方法：afterPropertiesSet()。在spring 初始化后，执行完所有属性设置方法(即setXxx)将自动调用 afterPropertiesSet().

public interface InitializingBean

{

public abstract void afterPropertiesSet()

throws Exception;

}

Bean实现这个接口，在afterPropertiesSet()中编写初始化代码即可。

2.一但bean实现了InitializingBean接口，那么这个bean的代码就和Spring耦合到一起了。通常情况下我不鼓励bean直接实现InitializingBean，可以使用Spring提供的init-method的功能来执行一个bean 子定义的初始化方法。

写一个java class，这个类不实现任何Spring的接口。定义一个没有参数的方法init()。然后在定义bean的时候指定初始化函数即可。

<bean name="lifeBean" class="research.spring.beanfactory.ch4.LifeCycleBean"

 init-method="init"><bean>

3. BeanPostProcessor接口，该接口作用是：如果我们需要在Spring容器完成Bean的实例化，配置和其他的初始化后添加一些自己的逻辑处理.

// Bean 实例化**之前**进行的处理

**public** Object postProcessBeforeInitialization(Object bean, String beanName)**throws** BeansException {

       System.*out*.println("对象" + beanName + "开始实例化");

**return** bean;

    }

    // Bean 实例化**之后**进行的处理

**public** Object postProcessAfterInitialization(Object bean, String beanName)**throws** BeansException {

       System.*out*.println("对象" + beanName + "实例化完成");

**return** bean;

    }