beanDefinitionMap:存放beanName对应的BeanDefinition singletonObjects:存放单例池 BeanDefinition: 对 beanPostProcessorList: 存放BeanPostProcessor bean的定义 拿到配置类上的注解 LysApplicationContext applicationContext = 扫描指定路径下的类 configClass.getAnnotation new Lys Application Context (App Config. class) scan(configClass); (ComponentScan.class) Class clazz = componentScanAnnotation.value(); beanDefinition.getType(); 拿到扫描路径 遍历 beanDefinitionMap UserInterface userService = (UserInterface) 对扫描路径进行处理,得 applicationContext.getBean("userService"); 到扫描路径下的所有.class 判断作用域是否为单 的absolutePath instance = clazz.getConstructor().new 例 Instance();//调用无参构造方 法创建对象 通过反射拿到Class classLoader.loadClass(absolutePath) createBean(beanNa 判断是否是单 创建单例bean me, bean Definition) 例bean createBean(beanName beanDefinition) 遍历每个属性 进行依赖注入 判断类上是否有 Component注解 从单例池中拿 放进单例池 到bean field.is Annotation Pres singletonObjects 判断是否有类实现了 ent(Autowired.class) BeanPostProcessor接口,将实现 BeanPostProcessor接口的类放进 检查是否需要依赖注入 beanPostProcessorList field.setAccessible(true); 拿到bean的 field.set(instance,getBean(field.getName())); beanName,作用域等 信息,并生成 BeanDefinition 如果有实现Aware接 口,执行实现Aware 接口的类中的方法 将生成的BeanDefinition放入beanDefinitionMap beanDefinitionMap.put(beanName,beanDefinition) 初始化前 bean Post Processor. post Process Befor eInitialization(instance,beanName) 初始化 ((InitializingBean) instance).afterPropertiesSet() 这步可以实现AOP功能,返回 初始化后 一个代理对象,如果不用实现 bean Post Processor. post Process After

AOP就返回一个原对象

nitialization(instance,beanName)

创建容器

获取bean