使用maven 管理项目工程

使用注解代替xml配置

**注解：**

@Configuration，配置类

@Component

@ComponentScan(..,includeFilters={@Filter},)扫描指定路径下的components

@Bean标记单个bean，可以指定id，默认为首字母小写的方法名，还可以指定(initMethod="init", destroyMethod="destroy")，JSR250规则定义的(java规范，非spring)两个注解@PostConstruct与@PreDestroy可以实现同样的功能

@Scope(“prototype”) prototype:多实例（容器初始时不加载）; singleton:单实例;request; session

@Lazy容器初始时不加载，取时才实例化放入容器

@Conditinal(XXCondition.class)XXCondition.java类做为条件类, 同时必须得实现spring提供的Confition接口

@Import，可以import指定的类，类数组，或者实现ImportSelector接口selectImports方法返回的类数组，还可以通过ImportBeanDefinitionRegistrar自定义注册bean，直接向容器中注册bean

**FactoryBean 与BeanFactory**

BeanFactory是bean工厂，可以通过该类实例取bean

FactoryBean<?>相当于一个bean的泛型包装类

**bean的生命周期**

BeanPostProcessor类[interface]: bean的后 置处理器,在bean初始化之前调用进行拦截

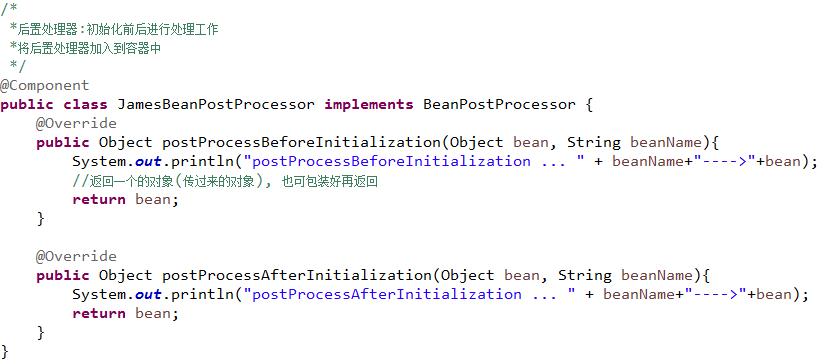
作用:在bean初始化前后进行一些处理工作, 打开此类

a> postProcessBeforeInitialization():在初始化之前进行后置处理工作(在init-method之前),

什么时候调用:它任何初始化方法调用之前(比如在InitializingBean的afterPropertiesSet初始化之前,或自定义init-method调用之前使用)

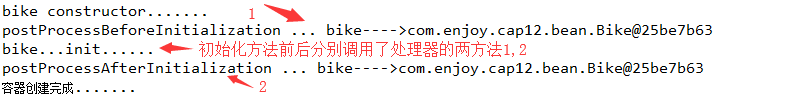
b> postProcessAfterInitialization():在初始化之后进行后置处理工作, 比如在InitializingBean的afterPropertiesSet()

步骤: 新建后置处理器类JamesBeanPostProcessor



总结:bean的整个生命周期我们都能控制

可以分析Bike的日志证明在init初始化之前调用了**postProcessBeforeInitialization(),**

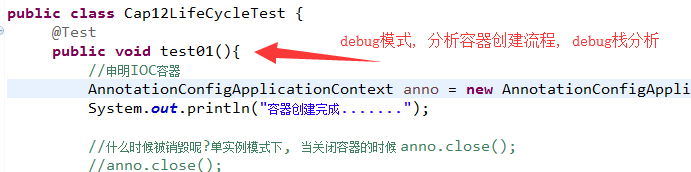


**容器启动及BeanPostProcessor源码分析**

针对以上cap12进行debug测试;



加完断点后, 测试用例debug, 分析容器创建流程



BeanPostProcessor原理:

可从容器类跟进顺序为:

AnnotationConfigApplicationContext-->refresh()-->

finishBeanFactoryInitialization(beanFactory)--->

beanFactory.preInstantiateSingletons()-->

760行getBean(beanName)--->

199行doGetBean(name, **null**, **null**, **false**)-->

317行createBean(beanName, mbd, args)-->

501行doCreateBean(beanName, mbdToUse, args)-->

541行createBeanInstance(beanName, mbd, args)(完成bean创建)-->

578行populateBean(beanName, mbd, instanceWrapper)(属性赋值)-->

579行initializeBean(beanName, exposedObject, mbd)(Bean初始化)->

1069行到1710行,后置处理器完成对init方法的前后处理.

最终得到如下如下

createBeanInstance(beanName, mbd, args)(完成bean创建)

populateBean(beanName, mbd, instanceWrapper); 给bean进行属性赋值

initializeBean() //初始化Bean方法内容如下,后置处理器对init方法的前后处理

{

applyBeanPostProcessorsBeforeInitialization(wrappedBean, beanName);

**invokeInitMethods(beanName, wrappedBean, mbd)** //执行自定义初始化

applyBeanPostProcessorsAfterInitialization(wrappedBean, beanName)

}

从以上分析不难发现,bean的生命周期为bean的创建, 初始化, 当容器关闭时对单实例的bean进行销毁.