Spring:IOC AOP

IOC 控制反转：用来解决层层之间 耦合

A B
@AutoWired
IB b;

Spring ioc(A ,B)

BeanFactory Spring顶层核心接口,使用了简单工厂模式，负责生产Bean

BeanDefinition Spring顶层核心接口,封装了生产Bean的一切原料

BeanFactoryPostProcessor:对bean定义做修改或者注册

BeanPostProcessor:对bean生产过程做扩展

1.配置类 xml @注解 javaconfig
2.加载spring上下文
xml: new ClassPathXmlApplicationContext("xml");
@: new AnnotationConfigApplicationContext(config.class)
3.getBean()

自定义类，实现BeanFactoryPostProcessor，重写postProcessorBeanFactory方法，该方法有BeanFactory参数，可以获得bean定义，然后自己可以对相应bean定义进行相应修改

注：
BeanDefinitionRegistryPostProcessor是BeanFactoryPostProcessor的子接口

自定义类，实现BeanDefinitionRefistryPostProcessor，重写postProcessorBeanDefinitionRegistry方法，该方法拿到BeanDefinitionRegistry参数，自己就可以在这里操作
比如，多注册一个bean等

两个重要扩展节点，在applicationContext中完成

扩展接口，Bean工厂的后置处理器，可以修改Bean定义
当spring集成其他框架时会用到下面两个扩展接口

BeanFactoryPostProcessor

BeanDefinitionRegistryPostProcesso r

修改

注册

BeanFactory

BeanDefinitionMap

BeanDefinition bean定义

类

xml

@注解

事件

ClassPathXmlApplicationContext

AnnotationConfigApplicationContext

BeanDefinitionReader 读出配置类信息

BeanDefinitionSacnner 扫描出有用的类

BeanDefinitionRegistry 把封装成BeanDefiniton

getBean

在这之前属于材料准备，接下去就是真正生产bean的地方

BeanPostProcessor

bean生命周期

Bean

BeanPostProcessor
实例化
反射

BeanPostProcessor

BeanPostProcessor

填充属性
@Autowired @Value

BeanPostProcessor
初始化
initMethod destroy

BeanPostProcessor

put

Map<beanname,bean实例>
单例池
一级缓存

在bean的生命周期中，比如实例化前后
属性填充前后、初始化前后等会调用
共9次的Bean后置处理器，这些都是扩展接口
扩展点

BeanFactory会实现一大堆生命周期Aware接口，在初始化时调用
怎么玩？
bean实现相应的Aware接口，则在初始化时会调用相应的实现方法

Aware

Aop

CGLIB

jdk

实例化

反射

工厂

比如注解、xml就是spring控制的来生成bean

javaConfig配置的，生产bean就是自己可以控制，可以直接操作new

填充属性

三级缓存解决循环依赖问题

B依赖A

class BeanA {
@Autowired
BeanB b;
}

class BeanB {
@Autowired
BeanA a;
}

A依赖B

三级缓存 三个Map

BeanFactory和ApplicationContext的异同点：
相同：
都有生产bean能力（ApplicationContext实现了BeanFactory）
异同：个人理解
BeanFactory生产Bean比较傻逼，功能比较简单，一步一步的需要手动去组装
ApplicationContext比较智能，可以理解为智能化封装好
从代码角度：
BeanFactory只能根据BeanDefinition通过getBean生产Bean
Application则智能化，能根据配置文件读取配置，再封装成BeanDefinition，注册BeanDefinition,然后再根据getBean生产出Bean

搞装修 柜子

工厂
BeanFactory

衣柜@Bean

getBean("user")

map.get()

图纸
BeanDefinition

衣柜店
XMLApplicationContext

设计设
BeanDefiniti
onRegistry

@Lazy
@Scope

颜色
款式

你
@Component
class

橱柜店
@ApplicationContext

设计设
BeanDefiniti
onRegistry

接待
扫描
BeanDefinitionSacnner

@Component

100

10

销售
BeanDefinitionReader

楼盘
配置类
.xml