Spring基础课程
强调 单实例对象 ----> 工厂启动的时候 被创建

Spring源码分析
现实 单实例对象 ----> getBean方法的时候现创建

ApplicationContext
ClassPathXmlApplicationContext
AnnotationConfigApplicationContext
DefaultListableBeanFactory

client
ApplicationContext
单实例对象 预先创建【默认】
DefaultListableBeanFactory

Spring创建对象过程中 分阶段创建
1. 创建对象
2. 属性的填充（注入）
3. 初始化的工作

其他Scope

scope Spring 5种
scope = Singleton | Prototype | request | session| globalSession 【SSO】

scope=request 底层实现 request.setAtrributes()
session.setAttributes()

ThreadLocal.set(object) ---Key thread Value object

回顾前面讲过的内容

```
<bean id="userDAO" class="xxxx.UserDAOImpl" scope="prototype"/>

<bean id="userService" class="xxx.UserServiceImpl" scope="singleton">
    <property name="userDAO" ref="userDAO"/>
</bean>
```

创建BeanDefinition

UserService us1 = beanFactory.getBean("userService")
UserService us2 = beanFactory.getBean("userService");

us1
userDAO1
us2
userDAO2

按道理说 userDAO1 != userDAO2 原因 scope=prototype

实际上 userDAO1 = userDAO2 结论 注入过程中 scope=prototype "失效 一个效果 "
BeanFactoryAware
UserService ----> beanFactory.getBean("userDAO")

prototype
缓存（错误）

getBean
单例池
创建对象
实例化对象
填充属性
beanFactory.getBean("userDAO")
userService
private UserDAO userDAO = beanFactory.getBean("userDAO")

重点的内容
循环引用的问题

结论
两个对象都是单实例的情况下，且通过set方式进行注入 才能成功
两个对象都是单实例的情况下，通过构造方法进行注入 不行
两个对象都是多实例的情况下 ，不管使用set或者构造 不行

A Proxy
A
Proxy B b
B Proxy
B
Proxy A a

getBean("a")
getSingleton() --> 不能A
doCreateBean
实例化-->创建a(b=null)
填充属性
getBean("b")
getSingleton() --> 不能B
doCreateBean
实例化 -->创建b(a=null)
填充属性
getBean("a") ?

初始化
b代理的创建

A(B Proxy)

初始化
A会不会做代理呢？

能否获取a?

1. 能获取 问题 从哪里获取？
singletonObjets
问题 此时 单例池 有？a

2. 不能获取 遇到什么问题？
getBean("a") 只能创建 新a对象

没有创建完全的a也得存储 ，但是不能存储singletonObjects里面

B注入a能不能获得代理
如何在这个位置 B获得a代理对象

代理缓存
a

DefaultListableBeanFacotry

singletonObjects
实例化 属性填充 初始化

earlySingletonObjects

singletonFactories
实例化后 非完全体的对象

存储不完全体 a
设想 ： singletonFactories "a" 不完全体的 A对象
实际 ： "a" getEarlyBeanReference() ---> BeanPostProccesor

之所以不存 不完全体的 A对象 而存一个方法。
方法意味 处理一些复杂功能？

addSingletonFactory(beanName, objectFactory);
addSingletonFactory(beanName, () -> getEarlyBeanReference(beanName, mbd, bean));
beanPostProcessor
调用过程中创建代理 把对应代理对象 放置 代理缓存池中

this.singletonObjects.get(beanName); -->null
this.earlySingletonObjects.get(beanName); -->null
this.singletonFactories.get(beanName);-->null

getBean("a")
getSingleton() --> 不能A
doCreateBean
实例化-->创建a(b=null)
addSingletonFactory(a, () -> getEarlyBeanReference(beanName, mbd, bean));
填充属性
getBean("b")
getSingleton() --> 不能B
doCreateBean
实例化 -->创建b(a=null)
addSingletonFactory(b, () -> getEarlyBeanReference(beanName, mbd, bean));
填充属性
getBean("a") ?

B.proxyA
初始化
b代理的创建 ProxyB
proxyA

A(B Proxy)

初始化
A会不会做代理呢？

this.singletonObjects.get(beanName) !=null

singletonObject = singletonFactory.getObject();
this.earlySingletonObjects.put(beanName, singletonObject);
this.singletonFactories.remove(beanName);

singletonObject -> getEalyBeanReferencs()
ProxyA --- Proxy池子
A

singletonFactories
实例化后 非完全体的对象

a () -> getEarlyBeanReference(beanName, mbd, bean))
b () -> getEarlyBeanReference(beanName, mbd, bean))

singletonFactories
实例化后 非完全体的对象

b () -> getEarlyBeanReference(beanName, mbd, bean))

earlySingletonObjects
Proxy A

singletonObjects
Proxy B
Proxy A