

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.setAttributes()
session.setAttributes()

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

回顾前面讲过的内容

```
<bean id="userDAO" class="xxx.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

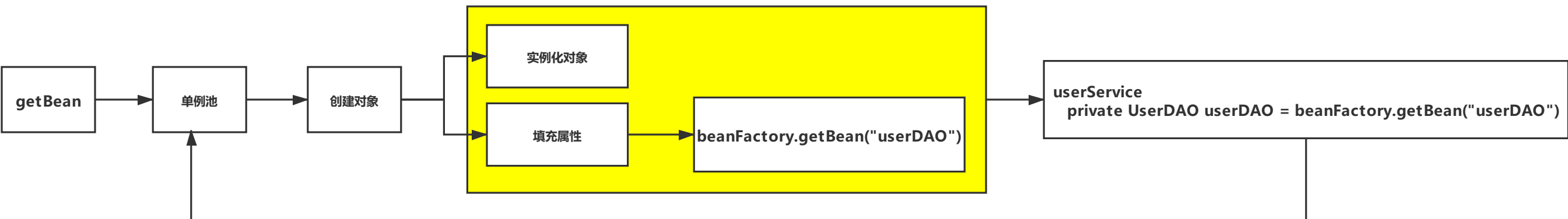
userDAO2

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）

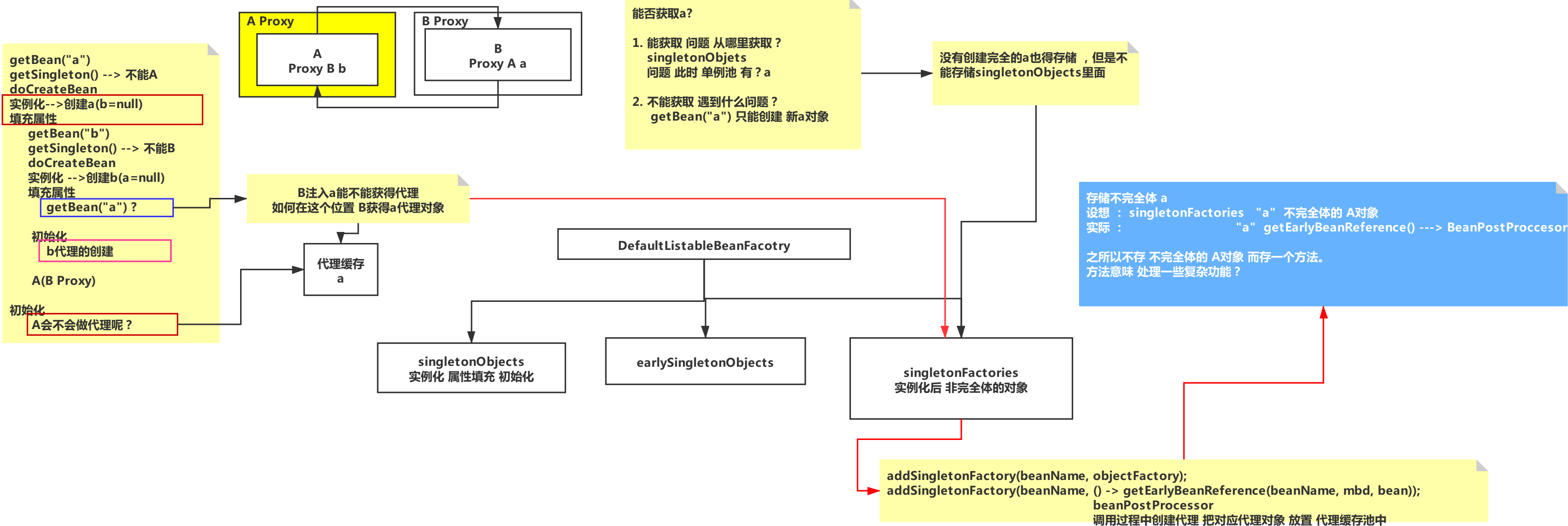


重点的内容

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



```
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
this.earlySingletonObjects.get(beanName); --> null
this.singletonFactories.get(beanName); --> null
this.singletonFactories.get(beanName) != null
singletonObject = singletonFactory.getObject();
this.earlySingletonObjects.put(beanName, singletonObject);
this.singletonFactories.remove(beanName);
```

singletonObject -> getEarlyBeanReferences()
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