1. 缓存
2. 内容

一级缓存：完整对象

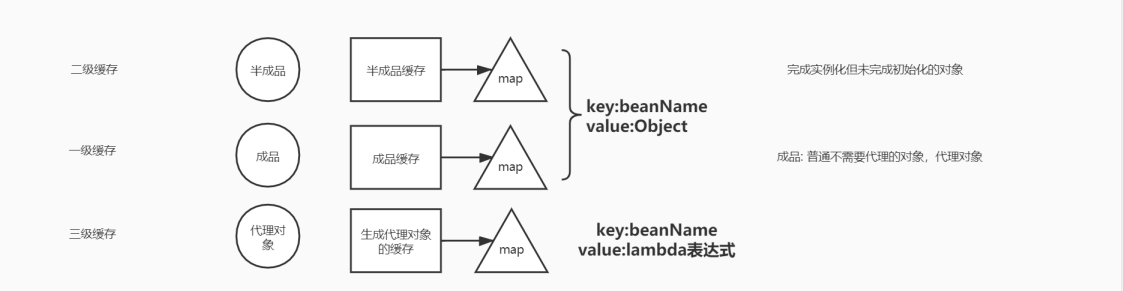
二级缓存：实例化未初始化对象

三级缓存：生成代理对象所需的匿名内部类、或者回调函数

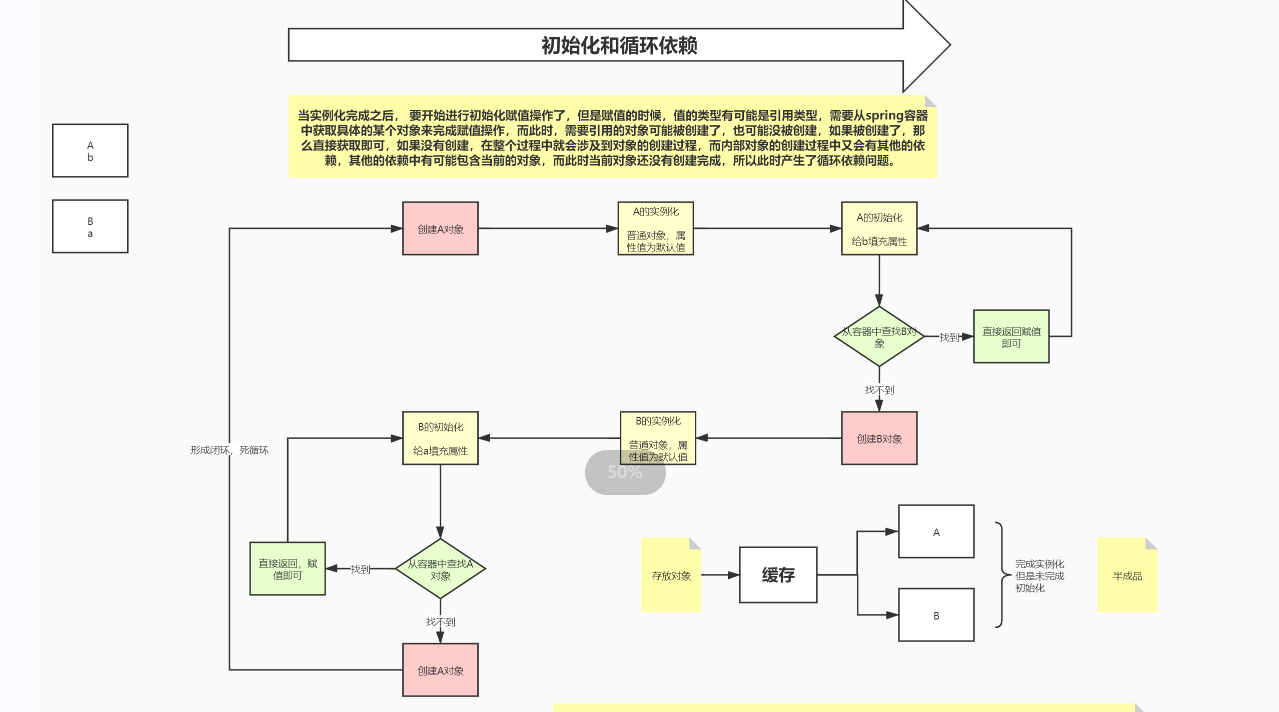
1. 作用

二级缓存是：为了解决循环依赖问题

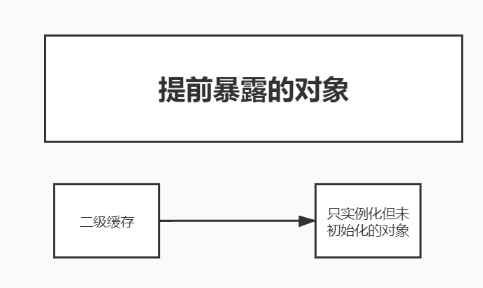
三级缓存是：为了代理（存放创建代理对象的Lambda表达式）

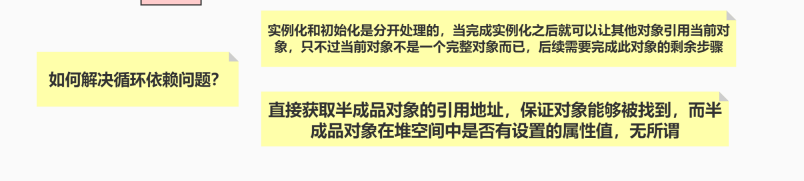


1. 循环依赖



1. 如何解决循环依赖？





1. 三级缓存中，对象只能存在一份，避免引用出错（代理对象、原始对象的名字一样）
2. Spring处理循环依赖

A->B->A : A调用getSingleton(此时一二级缓存没有A的信息，只有三级缓存有A的lambda表达式)，发现A正在创建过程中，且二级缓存中没有且允许被早期引用（即半成品就可以被引用），通过三级缓存的lambda即工厂创建一个对象，然后添加到二级缓存，并从三级缓存中移除lambda表达式。

最后创建出完整对象，加入一级缓存，并从二三级缓存中移除。

1. Spring解决代理对象的循环引用问题

解决代理对象（代理对象的名字于原始对象一样）：

A->B->A：

A需要代理

A先往三级缓存中添加lambda表达式（参数包含当前创建的bean对象）以便后续循环引用的时候，从三级缓存中取出lambda表达式创建A的代理对象，在填充B属性的时候还未创建代理对象

B引用A的时候，不能取A的原始对象，需要通过lambda表达式创建A的代理对象，因此，判断如果三级缓存有A的lambda表达式（生成代理对象）则创建A的代理对象，然后返回给B

1. 修改代码：（不需要代理，改为二级缓存）

主要是两处逻辑：

（1）doCreateBean的addSingletonFactory：加入三级缓存

（2）getSingleton(String beanName, boolean allowEarlyReference)：从缓存中获取对象