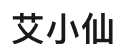


CodeSheep 2021-01-15 09:28

以下文章来源于艾小仙，作者艾小仙



一个愤世嫉俗，脱离低级趣味的人



往期置顶资源整理

- [数据结构和算法刷题笔记.pdf下载](#)
- [找工作简历模板集\(word格式\)下载](#)
- [Java基础核心知识大总结.pdf 下载](#)
- [68道C/C++常见面试题 \(含答案\) 下载](#)
- [23种设计模式学习笔记.pdf下载](#)
- [Java后端开发学习路线+知识点总结](#)
- [前端开发学习路线+知识点总结](#)
- [大数据开发学习路线+知识点总结](#)

- [C/C++开发\(后台\)学习路线+知识点总结](#)
- [嵌入式开发学习路线+知识点总结](#)

1、说说Spring里用了哪些设计模式？

单例模式：Spring 中的 Bean 默认情况下都是单例的。无需多说。

工厂模式：工厂模式主要是通过 BeanFactory 和 ApplicationContext 来生产 Bean 对象。

代理模式：最常见的 AOP 的实现方式就是通过代理来实现，Spring 主要是使用 JDK 动态代理和 CGLIB 代理。

模板方法模式：主要是一些对数据库操作的类用到，比如 JdbcTemplate、JpaTemplate，因为查询数据库的建立连接、执行查询、关闭连接几个过程，非常适用于模板方法。

2、谈谈对IOC和AOP的理解？实现原理是什么？

IOC 叫做控制反转，指的是通过Spring来管理对象的创建、配置和生命周期，这样相当于把控制权交给了Spring，不需要人工来管理对象之间复杂的依赖关系，这样做的好处就是解耦。在Spring里面，主要提供了 BeanFactory 和 ApplicationContext 两种 IOC 容器，通过他们来实现对 Bean 的管理。

AOP 叫做面向切面编程，他是一个编程范式，目的就是提高代码的模块性。Spring AOP 基于动态代理的方式实现，如果是实现了接口的话就会使用 JDK 动态代理，反之则使用 CGLIB 代理，Spring中 AOP 的应用主要体现在 事务、日志、异常处理等方面，通过在代码的前后做一些增强处理，可以实现对业务逻辑的隔离，提高代码的模块化能力，同时也是解耦。Spring主要提供了 Aspect 切面、JoinPoint 连接点、PointCut 切入点、Advice 增强等实现方式。

3、JDK动态代理和CGLIB代理有什么区别？

JDK 动态代理主要是针对类实现了某个接口，AOP 则会使用 JDK 动态代理。他基于反射的机制实现，生成一个实现同样接口的一个代理类，然后通过重写方法的方式，实现对代码的增强。

而如果某个类没有实现接口，AOP 则会使用 CGLIB 代理。他的底层原理是基于 asm 第三方框架，通过修改字节码生成成一个子类，然后重写父类的方法，实现对代码

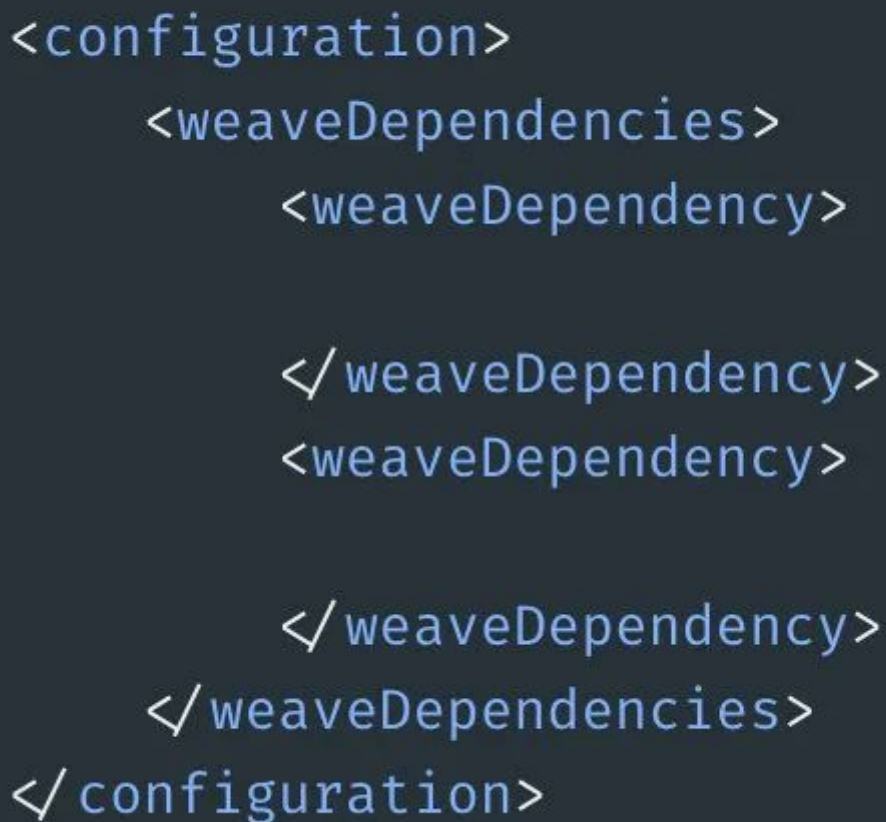
的增强。

4、Spring AOP 和 AspectJ AOP 有什么区别？

Spring AOP 基于动态代理实现，属于运行时增强。

AspectJ 则属于编译时增强，主要有3种方式：

1. 编译时织入：指的是增强的代码和源代码我们都有，直接使用 AspectJ 编译器编译就行了，编译之后生成一个新的类，他也会作为一个正常的 Java 类装载到JVM。
2. 编译后织入：指的是代码已经被编译成 class 文件或者已经打成 jar 包，这时候要增强的话，就是编译后织入，比如你依赖了第三方的类库，又想对他增强的话，就可以通过这种方式。



```
<configuration>
  <weaveDependencies>
    <weaveDependency>

    </weaveDependency>
    <weaveDependency>

    </weaveDependency>
  </weaveDependencies>
</configuration>
```

3. 加载时织入：指的是在 JVM 加载类的时候进行织入。

总结下来的话，就是 Spring AOP 只能在运行时织入，不需要单独编译，性能相比 AspectJ 编译织入的方式慢，而 AspectJ 只支持编译前后和类加载时织入，性能更好，功能更加强大。

5、FactoryBean 和 BeanFactory有什么区别？

BeanFactory 是 Bean 的工厂，ApplicationContext 的父类，IOC 容器的核心，负责生产和管理 Bean 对象。

FactoryBean 是 Bean，可以通过实现 FactoryBean 接口定制实例化 Bean 的逻辑，通过代理一个 Bean 对象，对方法前后做一些操作。

6、SpringBean的生命周期说说？

Spring Bean 生命周期简单概括为4个阶段：

1. 实例化，创建一个Bean对象

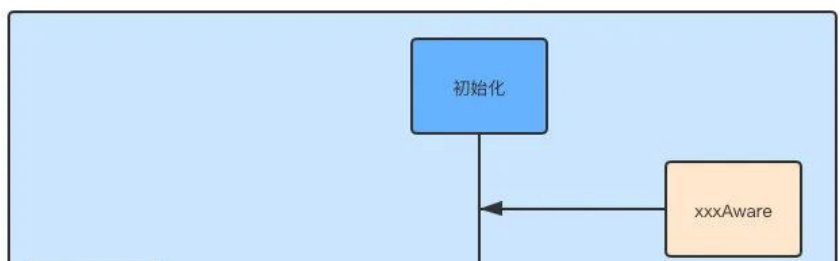
2. 填充属性，为属性赋值

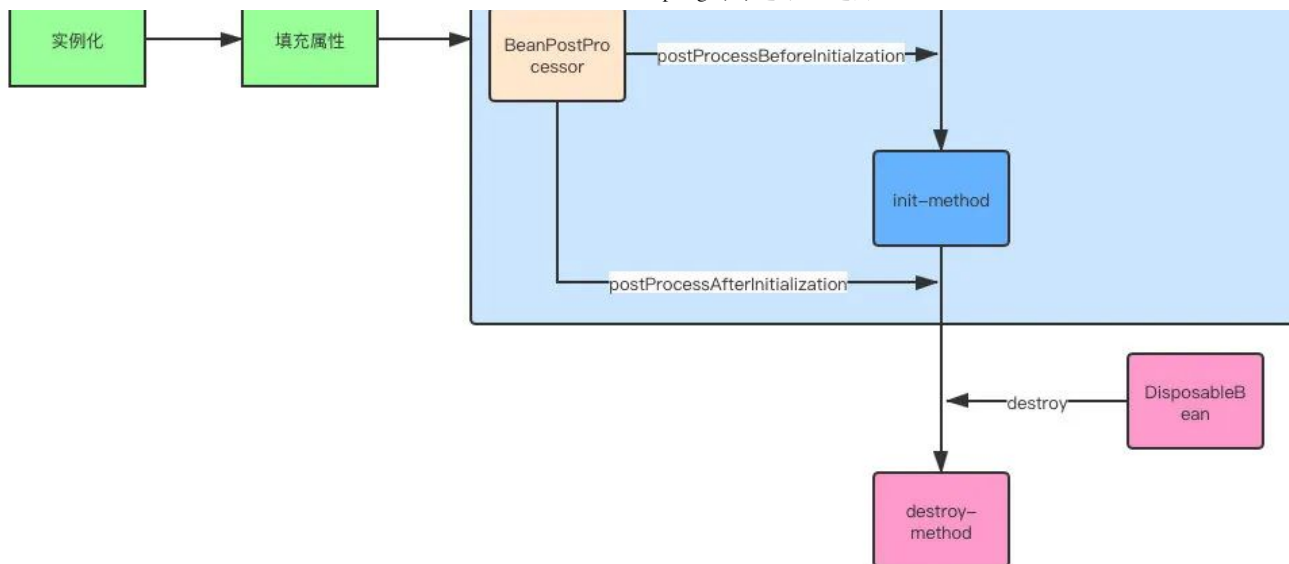
3. 初始化

- 如果实现了 `xxxAware` 接口，通过不同类型的Aware接口拿到Spring容器的资源
- 如果实现了BeanPostProcessor接口，则会回调该接口的 `postProcessBeforeInitialization` 和 `postProcessAfterInitialization` 方法
- 如果配置了 `init-method` 方法，则会执行 `init-method` 配置的方法

4. 销毁

- 容器关闭后，如果Bean实现了 `DisposableBean` 接口，则会回调该接口的 `destroy` 方法
- 如果配置了 `destroy-method` 方法，则会执行 `destroy-method` 配置的方法





7.Spring是怎么解决循环依赖的?

首先，Spring 解决循环依赖有两个前提条件：

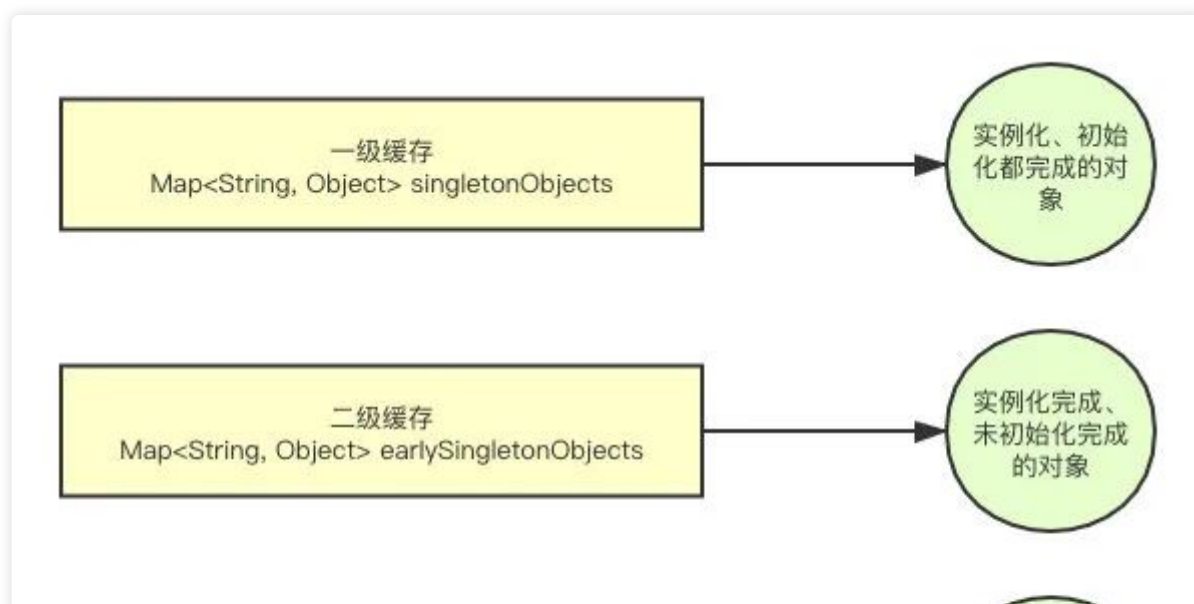
1. 不全是构造器方式的循环依赖
2. 必须是单例

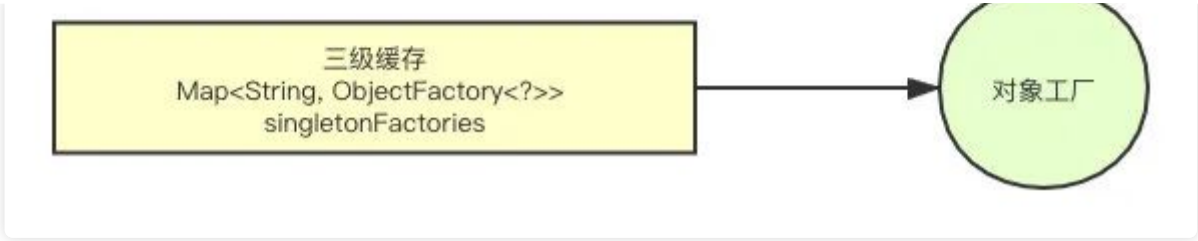
基于上面的问题，我们知道Bean的生命周期，本质上解决循环依赖的问题就是三级缓存，通过三级缓存提前拿到未初始化的对象。

第一级缓存：用来保存实例化、初始化都完成的对象

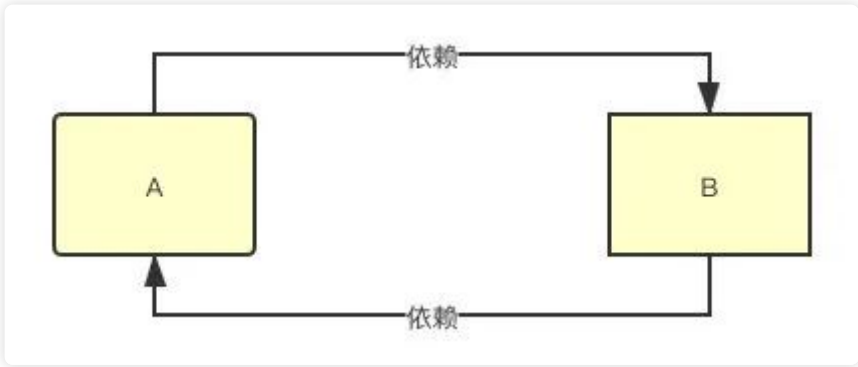
第二级缓存：用来保存实例化完成，但是未初始化完成的对象

第三级缓存：用来保存一个对象工厂，提供一个匿名内部类，用于创建二级缓存中的对象



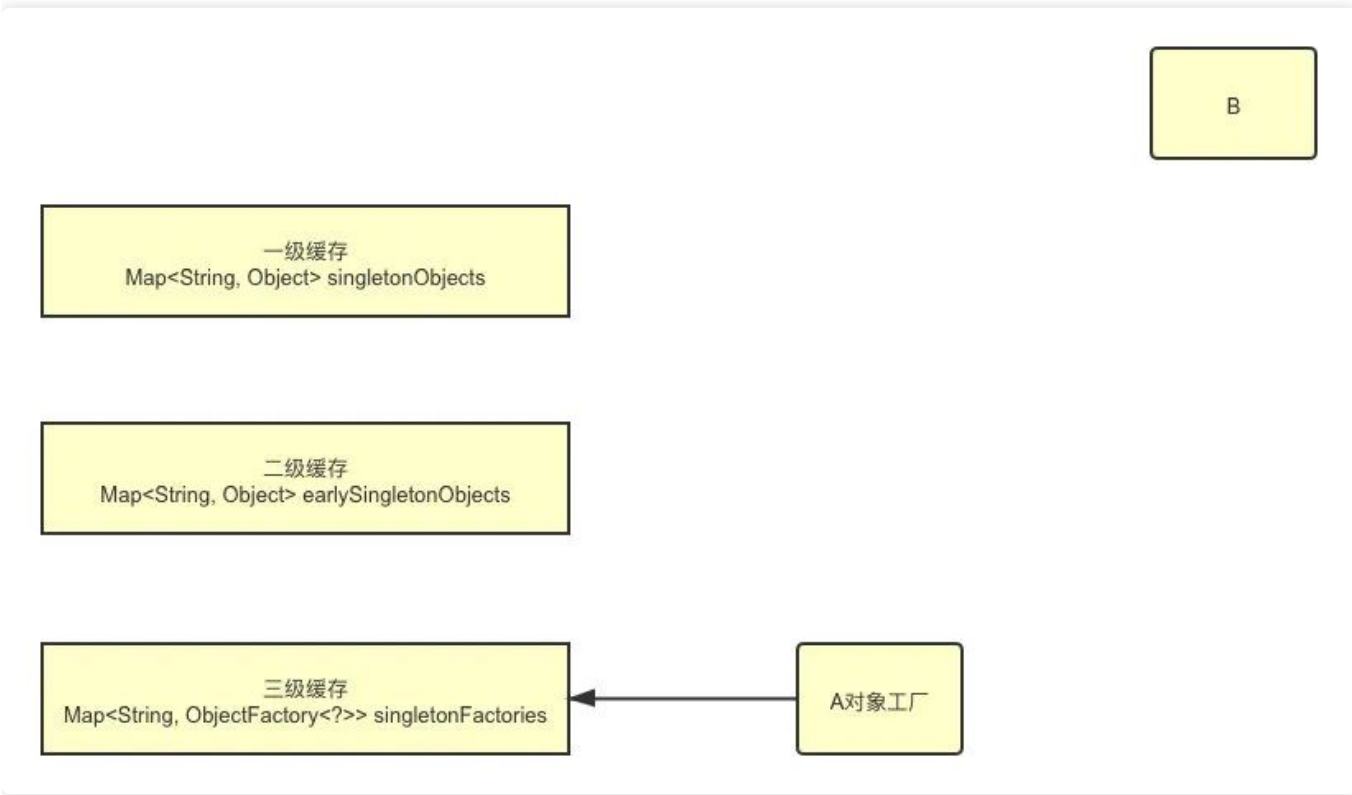


假设一个简单的循环依赖场景，A、B互相依赖。

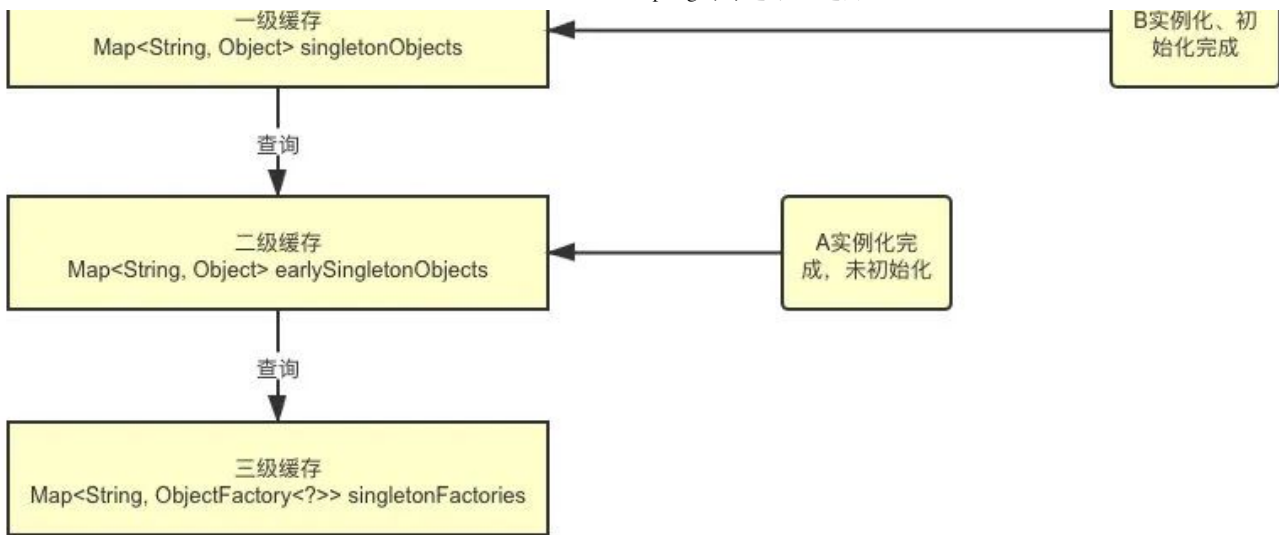


A对象的创建过程：

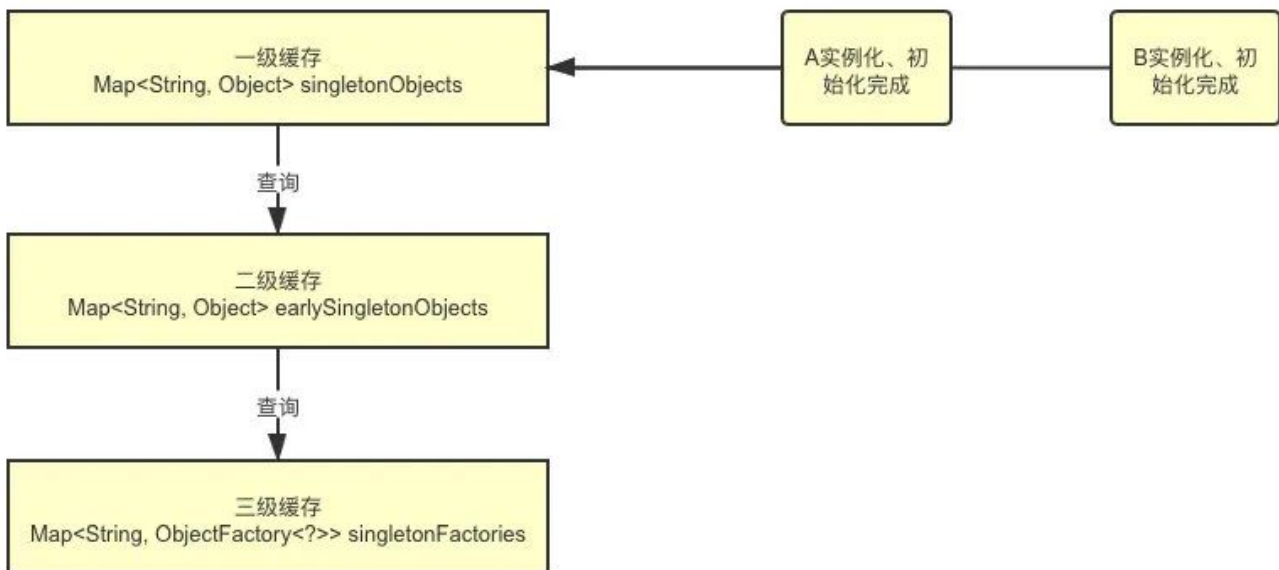
- 1. 创建对象A，实例化的时候把A对象工厂放入三级缓存



- 2. A注入属性时，发现依赖B，转而去实例化B
- 3. 同样创建对象B，注入属性时发现依赖A，一次从一级到三级缓存查询A，从三级缓存通过对象工厂拿到A，把A放入二级缓存，同时删除三级缓存中的A，此时，B已经实例化并且初始化完成，把B放入一级缓存。



- 接着继续创建A，顺利从一级缓存拿到实例化且初始化完成的B对象，A对象创建也完成，删除二级缓存中的A，同时把A放入一级缓存
- 最后，一级缓存中保存着实例化、初始化都完成的A、B对象



因此，由于把实例化和初始化的流程分开了，所以如果都是用构造器的话，就没法分离这个操作，所以都是构造器的话就无法解决循环依赖的问题了。

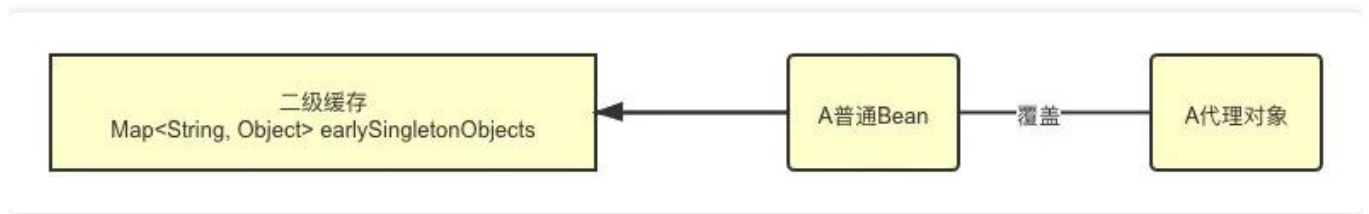
8、为什么要三级缓存？二级不行吗？

不可以，主要是为了生成代理对象。

因为三级缓存中放的是生成具体对象的匿名内部类，他可以生成代理对象，也可以是普通的实例对象。

使用三级缓存主要是为了保证不管什么时候使用的都是一个对象。

假设只有二级缓存的情况，往二级缓存中放的显示一个普通的Bean对象，**BeanPostProcessor** 去生成代理对象之后，覆盖掉二级缓存中的普通Bean对象，那么多线程环境下可能取到的对象就不一致了。



9、Spring事务传播机制有哪些？

1. **PROPAGATION_REQUIRED**：如果当前没有事务，就创建一个新事务，如果当前存在事务，就加入该事务，这也是通常我们的默认选择。
2. **PROPAGATION_REQUIRES_NEW**：创建新事务，无论当前存不存在事务，都创建新事务。
3. **PROPAGATION_NESTED**：如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则按REQUIRED属性执行。
4. **PROPAGATION_NOT_SUPPORTED**：以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。
5. **PROPAGATION_NEVER**：以非事务方式执行，如果当前存在事务，则抛出异常。
6. **PROPAGATION_MANDATORY**：支持当前事务，如果当前存在事务，就加入该事务，如果当前不存在事务，就抛出异常。
7. **PROPAGATION_SUPPORTS**：支持当前事务，如果当前存在事务，就加入该事务，如果当前不存在事务，就以非事务执行。'

10、最后，说说Spring Boot启动流程吧？

关于这个问题，之前专门写过一篇文章，可以参考：《[你知道Spring Boot是怎么启动的吗？](#)》。

这个流程，如果网上一搜，基本都是这张图了，我也不想再画一遍了。那其实主要的流程就几个大步骤：

1. 准备环境，根据不同的环境创建不同的Environment
2. 准备、加载上下文，为不同的环境选择不同的Spring Context，然后加载资源，配置Bean
3. 初始化，这个阶段刷新Spring Context，启动应用

[illegible]

图片来源于网络

- [数据结构和算法刷题笔记.pdf下载](#)
- [找工作简历模板大分享.doc下载](#)
- [Java基础核心知识大总结.pdf 下载](#)
- [68道C/C++常见面试题（含答案） 下载](#)
- [23种设计模式学习笔记.pdf下载](#)
- [Java后端开发学习路线+知识点总结](#)
- [前端开发学习路线+知识点总结](#)
- [大数据开发学习路线+知识点总结](#)
- [C/C++开发\(后台\)学习路线+知识点总结](#)
- [嵌入式开发学习路线+知识点总结](#)

弃用TCP