spring框架学习笔记

本笔记主要记录学习的知识，以供复习使用

1、IOC和AOP的概念

ioc(控制反转/反转控制)，它是一个技术思想，不是一个技术实现。描述的是Java开发领域对象的创建，管理的问题。传统开发方式中比如类A依赖于类B，往往会在类A中new一个B的对象，IoC思想下开发方式不用自己去new对象了，而是由IoC容器（Spring框架）去帮助我们实例化对象并且管理它，我们需要使用哪个对象，去问IoC容器要即可。即我们丧失了一个权利（创建、管理对象的权利）,得到了一个福利（不用考虑对象的创建、管理等一系列事情）。**我们把对象的创建和对象关系依赖交给了ioc容器来管理。**

AOP: Aspect oriented Programming 面向切面编程/面向方面编程。AOP是OOP的延续，OOP三大特征：封装、继承和多态，是一种垂直继承体系。在OOP的基础上增加横切逻辑的概念增强方法。横切逻辑代码存在一些问题：如横切代码重复问题，横切逻辑代码和业务代码混杂在一起，代码臃肿，维护不方便，AOP出场，AOP独辟蹊径提出横向抽取机制，将横切逻辑代码和业务逻辑代码分析。**AOP通过动态代理的方式在不改变业务逻辑的情况下增加功能。**

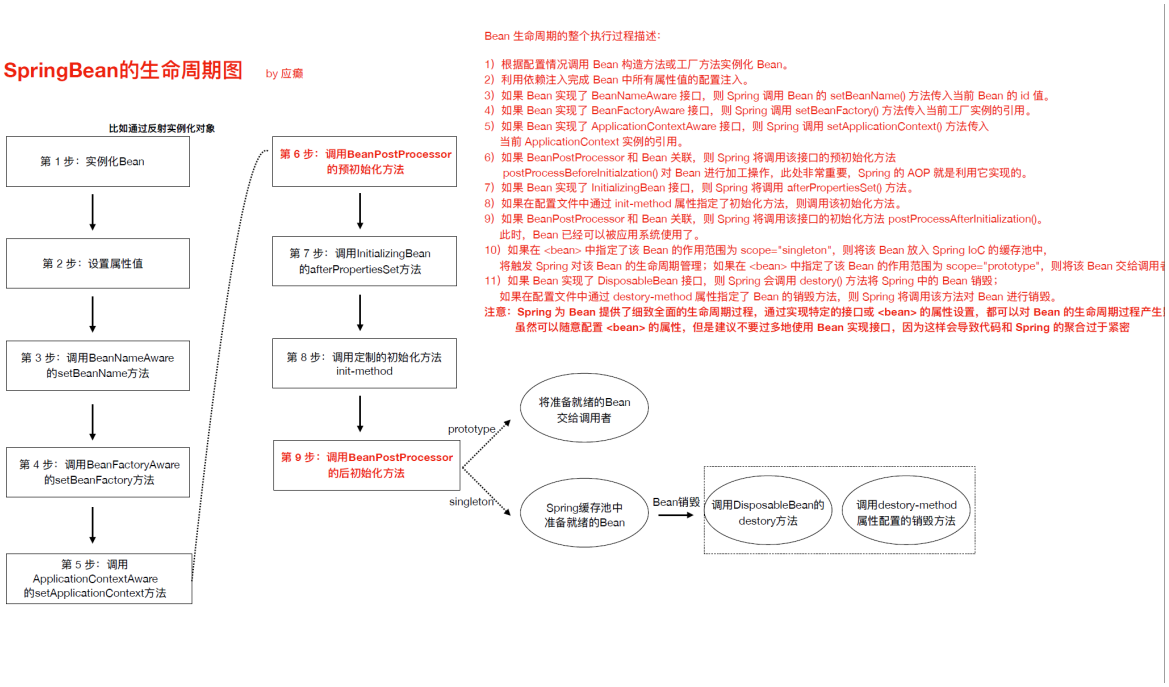
2、spring框架IOC和AOP应用的回顾

本部分为spring框架的基本使用，便于理解IOC和AOP的概念。

3、IOC源码剖析

IoC容器是Spring的核心模块，是抽象了对象管理、依赖关系管理的框架解决方案。Spring 提供了很多的容器，其中 BeanFactory 是顶层容器（根容器），不能被实例化，它定义了所有 IoC 容器必须遵从的一套原则，具体的容器实现可以增加额外的功能，比如我们常用到的ApplicationContext，更具体的实现如 ClassPathXmlApplicationContext 包含了解析 xml 等一系列的内容，AnnotationConfigApplicationContext 则是包含了注解解析等一系列的内容。Spring IoC 容器继承体系非常聪明，需要使用哪个层次用哪个层次即可，不必使用功能大而全的。

通过打断点的方式对Bean的创建进行源码剖析。重点理解IOC容器初始化流程，BeanFactory创建流程和Bean创建的流程。



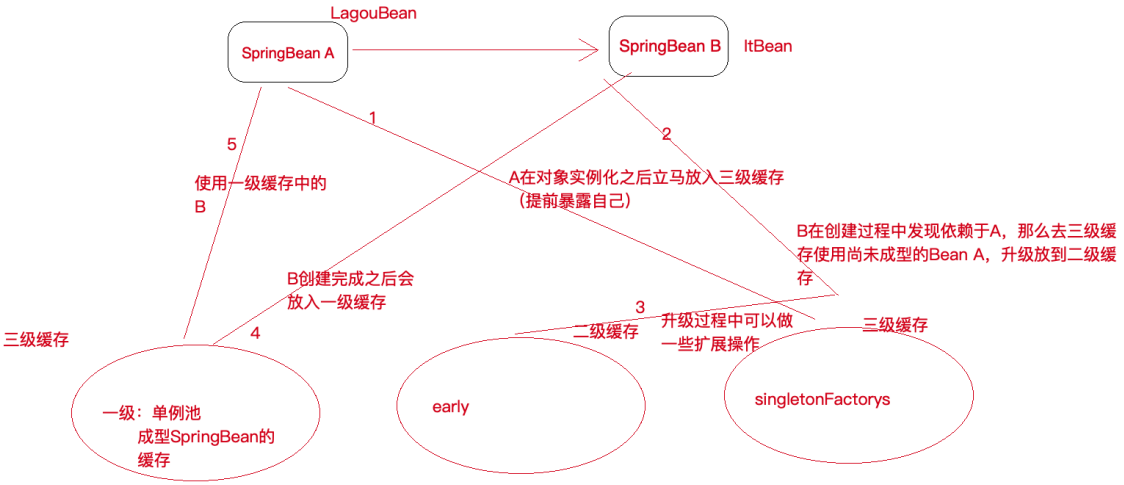
4、lazy-init延迟加载机制

普通 Bean 的初始化是在容器启动初始化阶段执行的，而被lazy-init=true修饰的 bean 则是在从容器里第一次进行context.getBean() 时进行触发。Spring 启动的时候会把所有bean信息(包括XML和注解)解析转化成Spring能够识别的BeanDefinition并存到Hashmap里供下面的初始化时用，然后对每个 BeanDefinition 进行处理，如果是懒加载的则在容器初始化阶段不处理，其他的则在容器初始化阶段进行初始化并依赖注入。

5、循环依赖问题

spring框架只能解决单例bean通过setXxx或者@Autowired进行循环依赖。

Spring 的循环依赖的理论依据基于 Java 的引用传递，当获得对象的引用时，对象的属性是可以延后设置的，但是构造器必须是在获取引用之前Spring通过setXxx或者@Autowired方法解决循环依赖其实是通过提前暴露一个ObjectFactory对象来完成的，简单来说ClassA在调用构造器完成对象初始化之后，在调用ClassA的setClassB方法之前就把ClassA实例化的对象通过ObjectFactory提前暴露到Spring容器中。



6、事务

- 掌握spring声明式事务控制的方法。
- 掌握事务的概念。
- 事务的隔离级别及产生的问题。

7、AOP源码剖析

通过AOP源码剖析掌握spring框架实现AOP的方式及注意事项。

重点理解代理对象的创建、Spring声明式事务控制的代码。