引言：循环依赖就是 N 个类中循环嵌套引用，如果在日常开发中我们用 new 对象的方式发生这种循环依赖的话程序会在运行时一直循环调用，直至内存溢出报错。下面说一下 Spring 是如果解决循环依赖的。

第一种：构造器参数循环依赖

Spring 容器会将每一个正在创建的 Bean 标识符放在一个“当前创建 Bean 池”中，Bean 标识符在创建过程中将一直保持在这个池中。

因此如果在创建 Bean 过程中发现自己已经在“当前创建 Bean 池”里时将抛出 BeanCurrentlyInCreationException 异常表示循环依赖；而对于创建完毕的 Bean 将从“当前创建 Bean 池”中清除掉。

首先我们先初始化三个 Bean。

```
public class StudentA {

    private StudentB studentB ;

    public void setStudentB(StudentB studentB) {
        this.studentB = studentB;
    }

    public StudentA() {
    }

    public StudentA(StudentB studentB) {
        this.studentB = studentB;
    }
}


public class StudentB {

    private StudentC studentC ;

    public void setStudentC(StudentC studentC) {
        this.studentC = studentC;
    }

    public StudentB() {
    }

    public StudentB(StudentC studentC) {
        this.studentC = studentC;
    }
}


public class StudentC {

    private StudentA studentA ;

    public void setStudentA(StudentA studentA) {
        this.studentA = studentA;
    }
```

```
    public StudentC() {
    }

    public StudentC(StudentA studentA) {
        this.studentA = studentA;
    }
}
```

OK，上面是很基本的 3 个类，，StudentA 有参构造是 StudentB。StudentB 的有参构造是 StudentC，StudentC 的有参构造是 StudentA ，这样就产生了一个循环依赖的情况，

我们都把这三个 Bean 交给 Spring 管理，并用有参构造实例化。

```
 <bean id="a" class="com.zfx.student.StudentA">
        <constructor-arg index="0" ref="b"></constructor-arg>
    </bean>
    <bean id="b" class="com.zfx.student.StudentB">
        <constructor-arg index="0" ref="c"></constructor-arg>
    </bean>
    <bean id="c" class="com.zfx.student.StudentC">
        <constructor-arg index="0" ref="a"></constructor-arg>
    </bean>
```

下面是测试类：

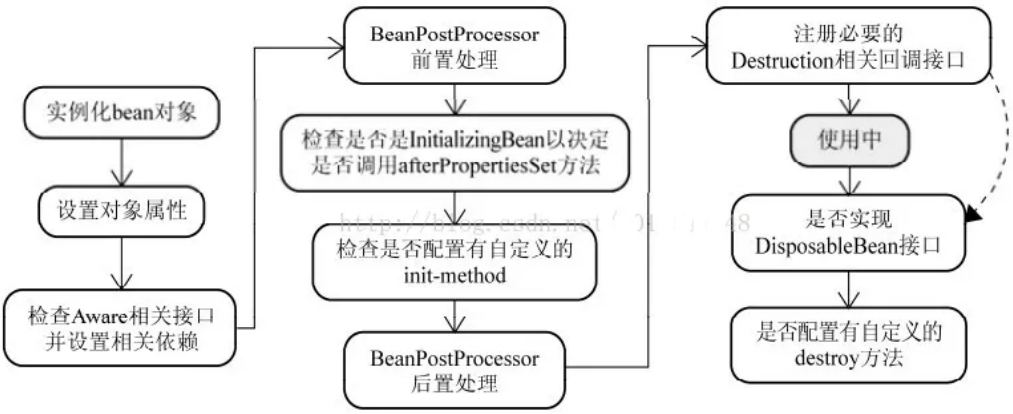
```
public class Test {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("com/zfx/student
/applicationContext.xml");
        //System.out.println(context.getBean("a", StudentA.class));
    }
}
```

执行结果报错信息为：

```
Caused by: org.springframework.beans.factory.BeanCurrentlyInCreationException:
    Error creating bean with name 'a': Requested bean is currently in creation: Is there
 an unresolvable circular reference?
```

如果大家理解开头那句话的话，这个报错应该不惊讶，Spring 容器先创建单例 StudentA，StudentA 依赖 StudentB，然后将 A 放在“当前创建 Bean 池”中，此时创建 StudentB,StudentB 依赖 StudentC ,然后将 B 放在“当前创建 Bean 池”中,此时创建 StudentC，StudentC 又依赖 StudentA， 但是，此时 Student 已经在池中，所以会报错，，因为在池中的 Bean 都是未初始化完的，所以会依赖错误 ，（初始化完的 Bean 会从池中移除）

第二种：setter 方式单例，默认方式



如果要说 setter 方式注入的话，我们最好先看一张 Spring 中 Bean 实例化的图

如图中前两步骤得知：Spring 是先将 Bean 对象实例化之后再设置对象属性的

修改配置文件为 set 方式注入

```
    <!--scope="singleton"(默认就是单例方式)  -->
    <bean id="a" class="com.zfx.student.StudentA" scope="singleton">
        <property name="studentB" ref="b"></property>
    </bean>
    <bean id="b" class="com.zfx.student.StudentB" scope="singleton">
        <property name="studentC" ref="c"></property>
    </bean>
    <bean id="c" class="com.zfx.student.StudentC" scope="singleton">
        <property name="studentA" ref="a"></property>
    </bean>
```

下面是测试类：

```
public class Test {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("com/zfx/student
/applicationContext.xml");
        System.out.println(context.getBean("a", StudentA.class));
    }
}
```

打印结果为：

```
com.zfx.student.StudentA@1fbfd6
```

为什么用 set 方式就不报错了呢 ？

我们结合上面那张图看，Spring 先是用构造实例化 Bean 对象 ，此时 Spring 会将这个实例化结束的对象放到一个 Map 中，并且 Spring 提供了获取这个未设置属性的实例化对象引用的方法。

结合我们的实例来看，，当 Spring 实例化了 StudentA、StudentB、StudentC 后，紧接着会去设置对象的属性，此时 StudentA 依赖 StudentB，就会去 Map 中取出存在里面的单例 StudentB 对象，以此类推，不会出来循环的问题喽、

下面是 Spring 源码中的实现方法。以下的源码在 Spring 的 Bean 包中的 DefaultSingletonBeanRegistry.java 类中

```
/** Cache of singleton objects: bean name --> bean instance（缓存单例实例化对象的Map 集合）
 */
    private final Map<String, Object> singletonObjects = new ConcurrentHashMap<String, O
bject>(64);

    /** Cache of singleton factories: bean name --> ObjectFactory（单例的工厂Bean 缓存集合）
 */
    private final Map<String, ObjectFactory> singletonFactories = new HashMap<String, Ob
jectFactory>(16);

    /** Cache of early singleton objects: bean name --> bean instance（早期的单身对象缓存
集合） */
    private final Map<String, Object> earlySingletonObjects = new HashMap<String, Object>(
16);

    /** Set of registered singletons, containing the bean names in registration order（单
例的实例化对象名称集合） */
    private final Set<String> registeredSingletons = new LinkedHashSet<String>(64);
    /**
     * 添加单例实例
     * 解决循环引用的问题
     * Add the given singleton factory for building the specified singleton
     * if necessary.
     * <p>To be called for eager registration of singletons, e.g. to be able to
     * resolve circular references.
     * @param beanName the name of the bean
     * @param singletonFactory the factory for the singleton object
     */
    protected void addSingletonFactory(String beanName, ObjectFactory singletonFactory)
{
        Assert.notNull(singletonFactory, "Singleton factory must not be null");
        synchronized (this.singletonObjects) {
            if (!this.singletonObjects.containsKey(beanName)) {
                this.singletonFactories.put(beanName, singletonFactory);
                this.earlySingletonObjects.remove(beanName);
                this.registeredSingletons.add(beanName);
            }
        }
    }
```

第三种：setter 方式原型，prototype

修改配置文件为：

```
<bean id="a" class="com.zfx.student.StudentA" scope="prototype">
        <property name="studentB" ref="b"></property>
    </bean>
    <bean id="b" class="com.zfx.student.StudentB" scope="prototype">
        <property name="studentC" ref="c"></property>
    </bean>
    <bean id="c" class="com.zfx.student.StudentC" scope="prototype">
        <property name="studentA" ref="a"></property>
    </bean>
```

scope="prototype" 意思是 每次请求都会创建一个实例对象。

两者的区别是：有状态的 bean 都使用 Prototype 作用域，无状态的一般都使用 singleton 单例作用域。

测试用例：

```
public class Test {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("com/zfx/student
/applicationContext.xml");
        //此时必须要获取 Spring 管理的实例，因为现在 scope="prototype" 只有请求获取的时候才会实例化对象
        System.out.println(context.getBean("a", StudentA.class));
    }
}
```

打印结果：

```
Caused by: org.springframework.beans.factory.BeanCurrentlyInCreationException:
    Error creating bean with name 'a': Requested bean is currently in creation: Is there
 an unresolvable circular reference?
```

为什么原型模式就报错了呢 ？

对于“prototype”作用域 Bean，Spring 容器无法完成依赖注入，因为“prototype”作用域的 Bean，Spring 容器不进行缓存，因此无法提前暴露一个创建中的 Bean。