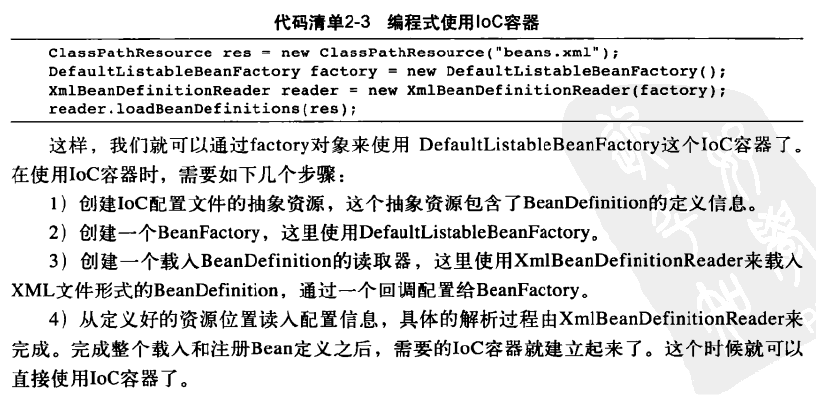
**Spring 通过定义BeanDefinition 来管理基于Spring 的应用中的各种对象以及它们之间的相互依赖关系, BeanDefinition抽象了我们对Bean的定义 , 是让容器器作用的主要数据类型. BeanDefinition就是对依赖反转模式中管理的对象依赖关系的数据抽象, 也是容器实现依赖反正功能的核心数据结构, 依赖反转功能都是围绕对这个BeanDefinition的处理来完成的**

Spring中的编程式使用ioc容器



**ApplicationContext 的继承关系**

ApplicationContext 的继承关系

|----ApplicationEventPublisher

|----HierarchicalBeanFactory

| |----BeanFactory

|----ListableBeanFactory

| |----BeanFactory

|----MessageSource

**Refresh**

Refresh方法启动包括BeanDefinition的Resource定位, 载入和注册三个基本的过程.

1 . Resource定位过程, 这个Resource定位指的是BeanDefinition的资源定位, 它由ResourceLoader通过统一的Resource接口来完成, 这个Resource对各种形式的BeanDefinition的使用都提供了统一接口.

2 . BeanDefinition的载入. 这个载入过程是把用户定义好的Bean表示成IoC容器内部的数据结构, 而这个容器内部的数据结构就是BeanDefiniton.

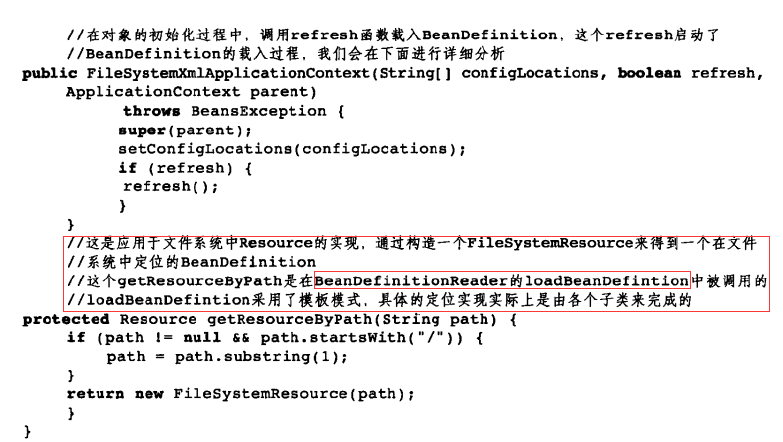
3. 向Ioc 容器注册这写BeanDefiniton的过程, 这个过程是通过调用向Ioc 容器进行注册. 在Ioc容器内部将BeanDefiniton注入到一个HashMap中去, Ioc容器就是通过这个HashMap来持有这些BeanDefiniton数据的. ( 在spring ioc的设计中 , Bean定义的载入和依赖注入是练个独立的过程. 依赖注入一般发生在应用第一次通过getBean向容器索取Bean的时候. 但有一个例外值得注意, 在使用Ioc容器时有一个预实例化的配置, 通过这预实例化的配置 (具体是通过为Bean定义信息中的lazyinit属性) 用户可以更嘎这个属性)

DefaultListtabeBeanFactory 只是一个纯粹的Ioc容器, 需要为它配置特定的读取器才能完成这些功能. 一般建议使用更高级的容器 如FileSystemXmlApplicationContext, ClassPathXmlApplicationContext 以及XmlWebApplicationContext等 .

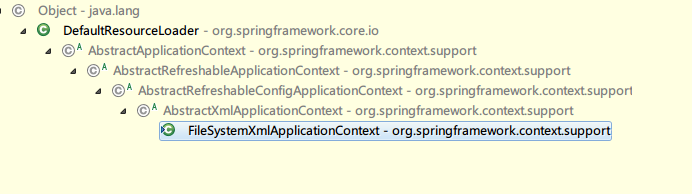
FileSystemXmlApplicationContext可以从文件系统载入Resource

ClassPathXmlApplicationContext可以从class path载入Resource

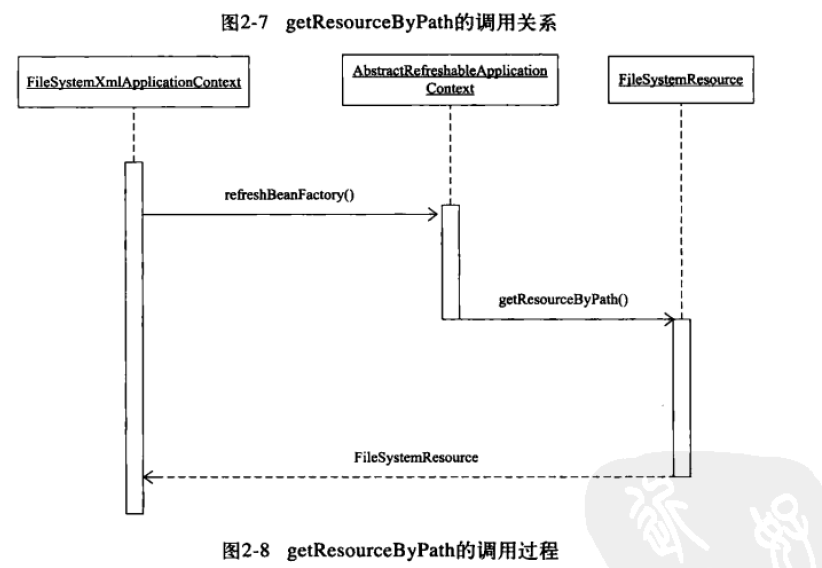
XmlWebApplicationContext 可以在Web容器中载入Resource

BeanDefinitionReader中的loadBeanDefinition方法

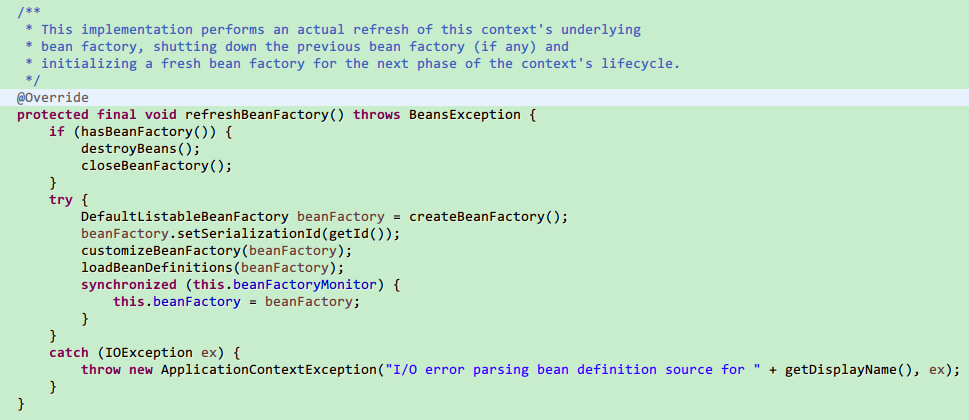
getResourceByPath 的继承结构



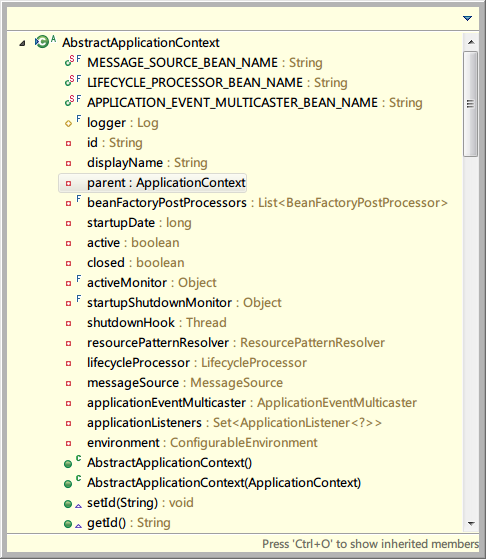
getResourceByPath的调用的关系



AbstractApplicationContext 中的 RefreshBeanFactiory() 方法

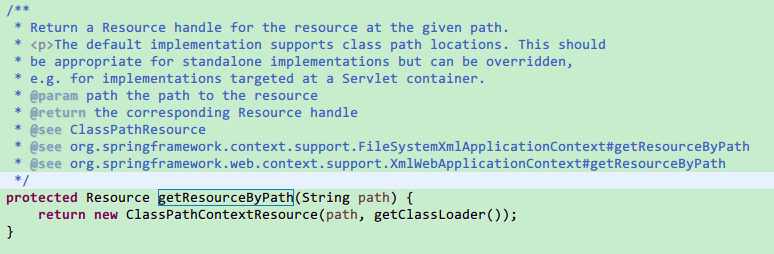


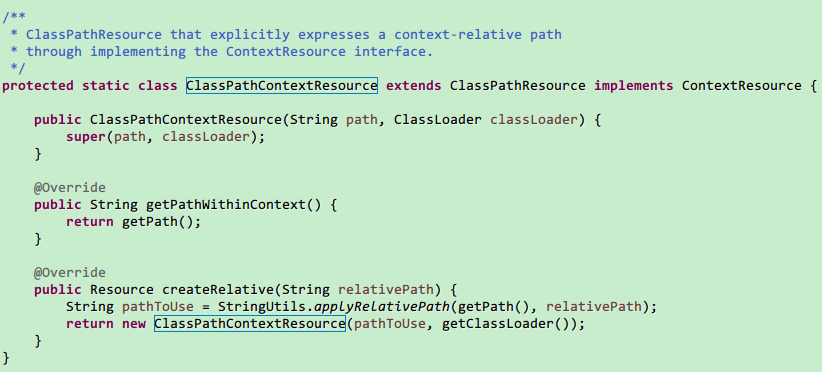
其容器的双亲在AbstractApplicationContext中的parent

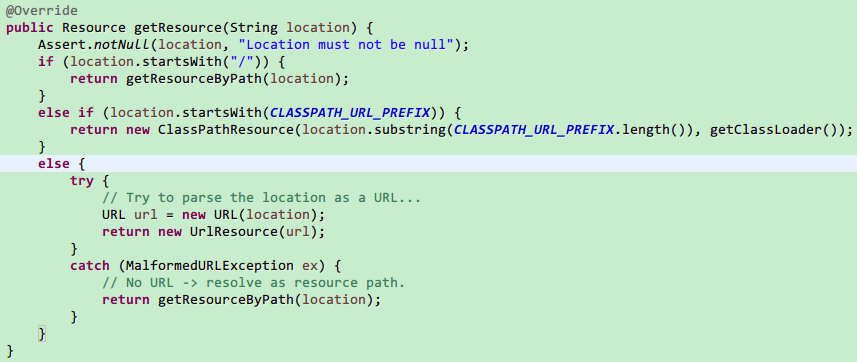


DefaultResourceLoader 中读取xml 获取Resource 获取的是classpathResource

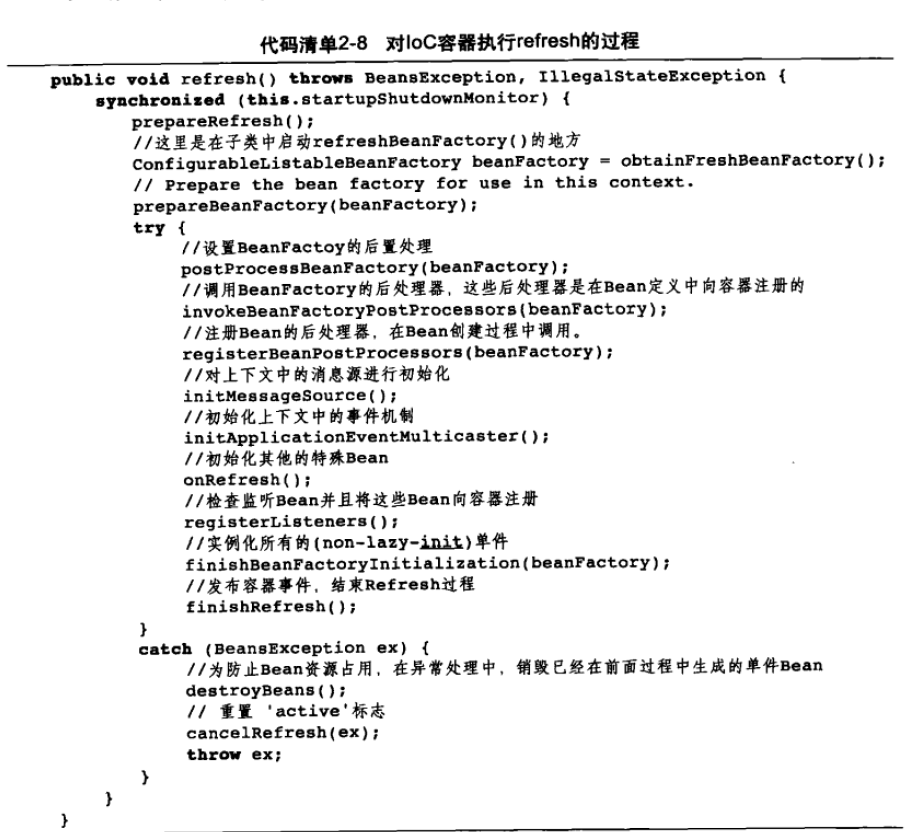
getResourceByPath 会被子类的FIleSystemXmlApplicatioContext实现

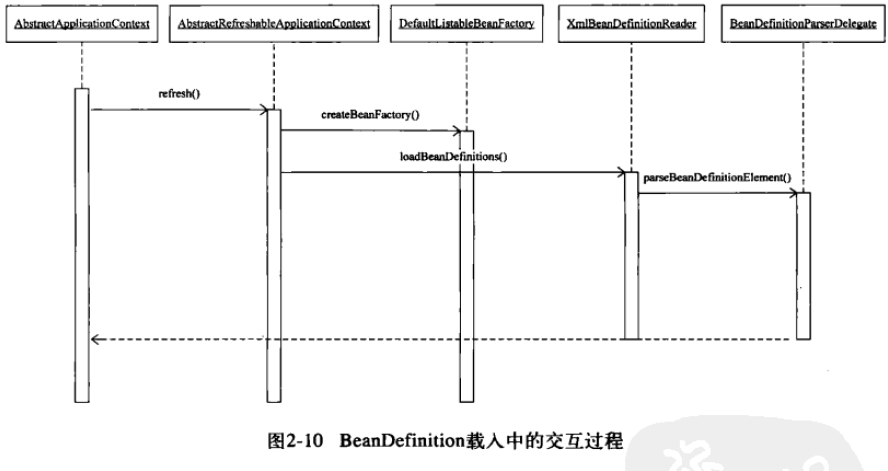
ClassPathContextResource类

getResource



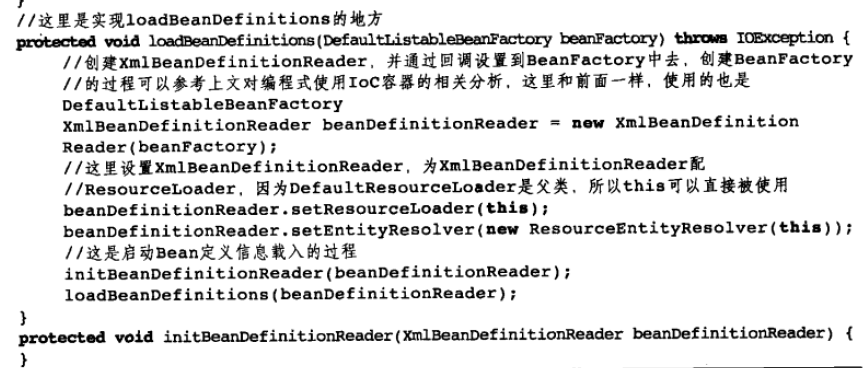
AbstractApplicationContext 中的Refresh方法



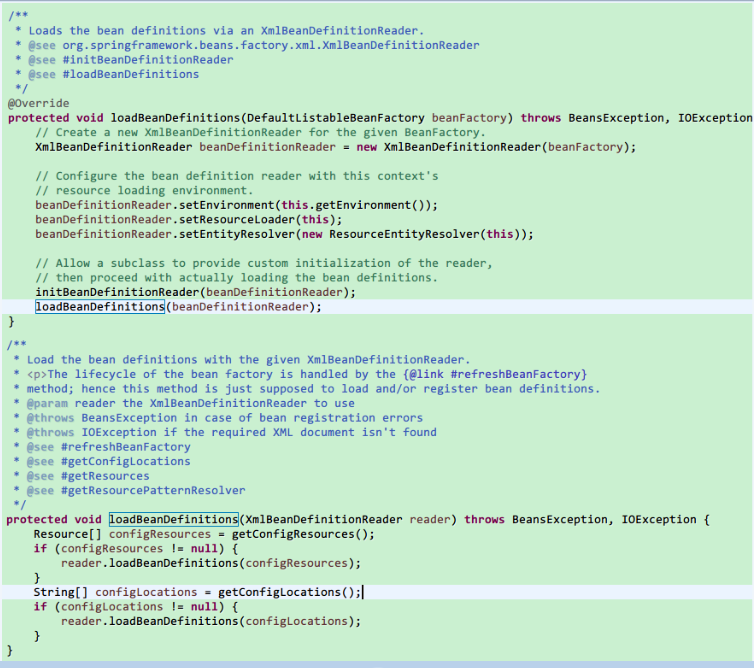


BeanDefinition载入中的

loadBeanDefinitions 是一个抽象方法 实际的载入过程在的AbstractRefreshableApplicaitonContext 的子类AbstractXmlApplicationContext中的实现, 其中初始化了读取器XmlBeanDefinitionReader, 然后把这个读取器在Ioc容器中设置好 最后是启动读取器来完成BeanDefinition在Ioc容器这的载入.



loadBeanDefinition的具体调用的处理



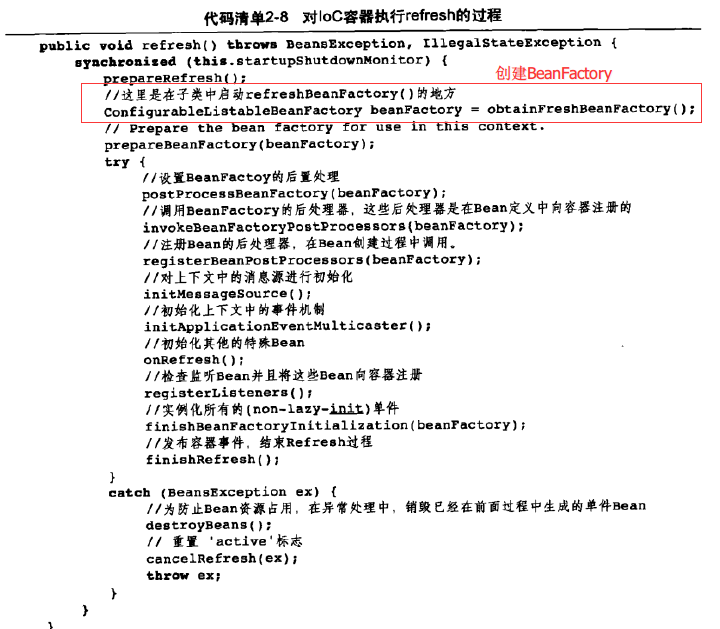
Resource 定位过程完成以后 为BeanDefinition的载入创造了I/O操作的条件, 但是还没有具体的读入数据 , 具体的读入数据是在BeanDefinition的载入和解析中来完成的.

**BeanDefinitioon的载入和解析**

BeanDefinition的载入过程就是把定义的BeanDefinition转成一个Spring 的内部数据结构的过程, IOC 容器对Bean的关联和依赖注入功能的实现 ,是通过对其持有的BeanDefinition进行各种相关操作来完成的, 这些BeanDefinition数据在ioc容器中通过一个HashMap来保持和维护.

A,具体如FileSystemXmlApplicationContext中的构造函数中的Refresh()是BeanDefinition的载入入口

B, refresh()方法是调用的是AbstractApplicationContext中的refresh()方法



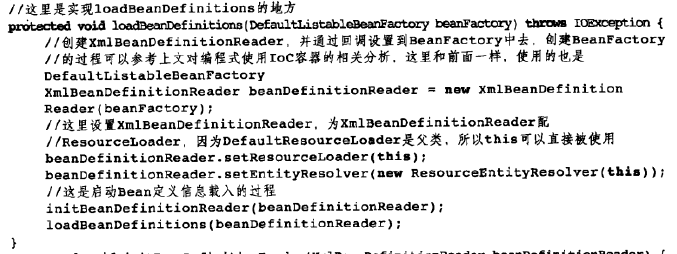
C, 在AbstractRefreshableApplicationContext的refreshBeanFactory()方法中创建BeanFactory

具体调用org.springframework.context.support.AbstractRefreshableApplicationContext

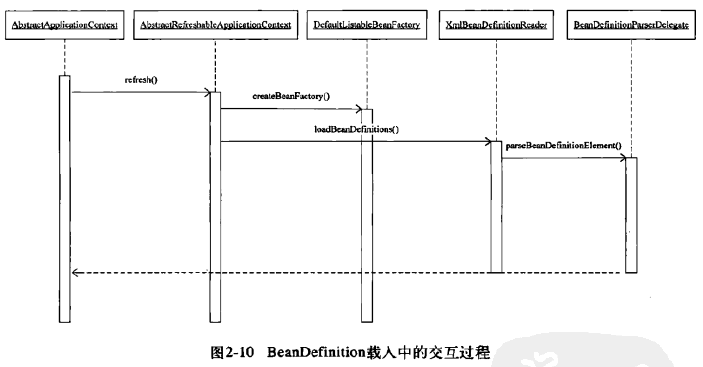
中的refreshBeanFactory()方法



在此中的loadBeanDefinitions(beanFactory);方法是抽象的方法 具体是在AbstractXmlApplicationContext中调用的



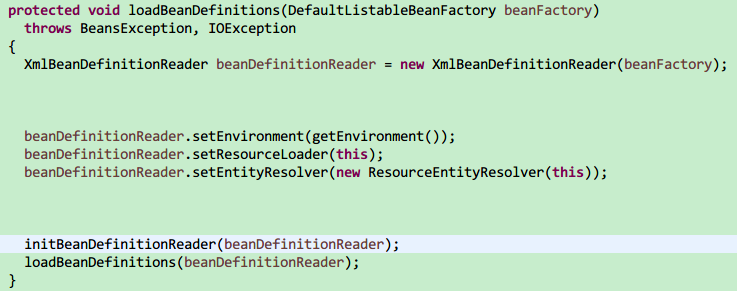
BeanDefinition载入的交互的过程



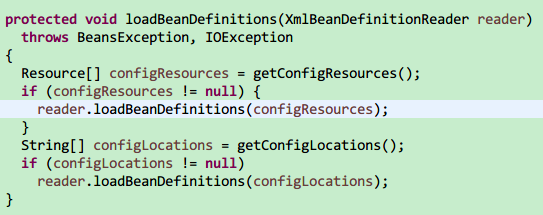
## BeanDefinition在IOC容器中的注册

**LoadBeanDefinition 代码 -- >**

**loadBeanDefinitions (XmlBeanDefinitionReader reader)**

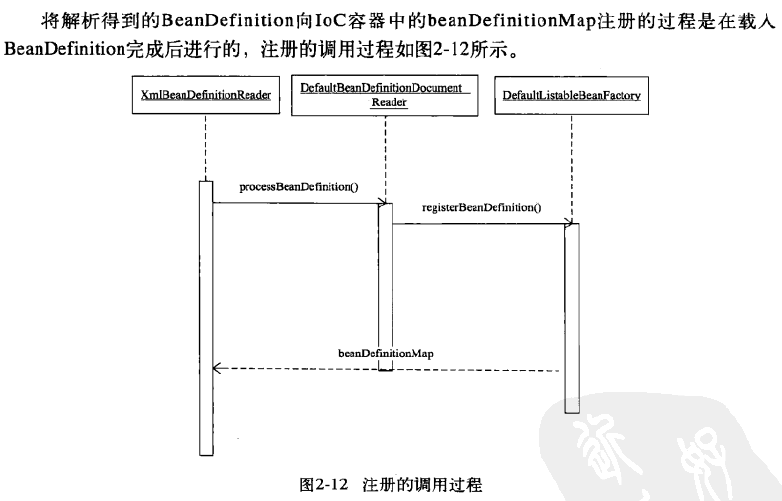


**LoadBeanDefinitions(XmlBeanDefinitionReader) 的代码**



**BeanDefinition 的载入分成两部分,首先通过调用xml的解析器得到document的对象,但这些document对象并没有按照spring的Bean规则进行解析, 在完成通用的xml解析以后.才是按照spring的Bean规则进行解析的. 这个spring的Bean规则进行解析的过程是在documentReader中实现的. 这里使用的documentReader是默认设置好的defaultBeanDefinitionDoucmentReader. 完成BeanDefinition的处理的结果是由BeanDefinitionHolder对象来持有的, 这个BeanDefinitionHolder的生成是通过对Document文档书的内容进行解析来完成的,这个解析的过程是由BeanDefinitionParserDelegate来实现的( 具体在processBeanDefinition方法中实现) , 解析过程与spring对BeanDefintion的配置规则紧密相关.**

**BeanDefinition 在ioc容器中的注册**



**IOC容器的依赖注入**