**目的: 为了使以下代码可以走通**

ApplicationContext ctx = new ClassPathXmlApplicationContext("petstore-v1.xml");

PetStoreService petStore = (PetStoreService)ctx.getBean("petStore");

**实现思路:**

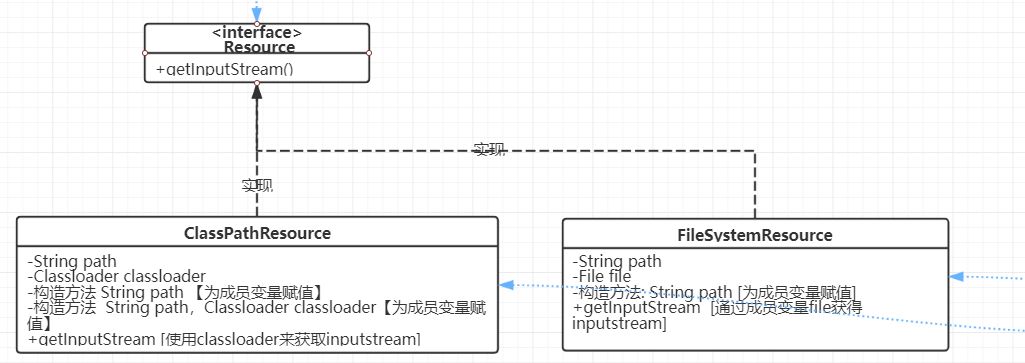
一.从xml中解析出bean的BeanDefinition (在利用反射创建bean的时候会用到类名, 从map中找到合适的BeanDefinition需要拿到key即beanId, 所以BeanDefiniton字段有beanId和beanClassName, 以及scope)

二.把BeanDefinition设置进入factory的成员变量Map<String,BeanDefinition> beanDefinitionMap中

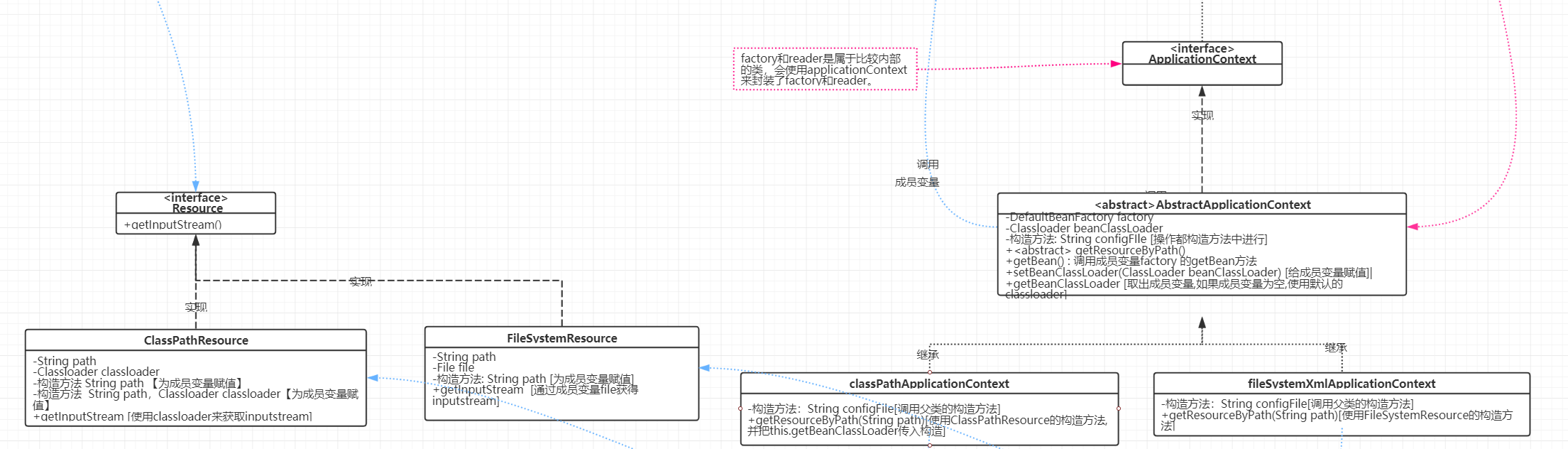
三.在getBean()的时候, 利用classLoader和类名. 创建一个实体bean

**拆解出几个点: 正叙**

1. 二. 涉及: 这个时候先不讨论DefaultBeanDefinition的复杂性. 只当做单一的一个类来看待
2. 在解析xml的时候, 需要读取xml文件, 最终要转化出inputStream流才能实现解析, 但是我们文件的来源有两种(这里只讲两种), 一种是文件在classPath路径下面, 另一种是文件在某个文件夹下面. 所以获得inputStream的方式也应该是两种, 他们既有相同的方法名称, 方法的实现方式还不一样. 把获取inputStream方法的类叫做Resource, 就有了两种Resource, 所以就有了下面的类的关系:



1. 既然Resourc有两种, 那么在获得Resource的时候也有两种选择. 所以我们的ApplicationContext也要有两种. 但是除了获得Resource的方法不同之外, 其他方法都是一样的, 所以就有了下面的类的关系. ----使用了抽象类写好了所有一致的方法

3. 现在我们已经获得了Resource了, 需要做的就是对这个Resource(获取resource代表的inputStream)进行解析, spring的职责是很明确的, 所以解析这个动作是单独做了一个类来完成的. 即XmlBeanDefinitionReader, 解析后得到的BeanDefinition是要放入factory的成员变量Map<String,BeanDefinition>的, 所以让XmlBeanDefinitionReader持有一个成员变量, factory.

具体的解析方法在XmlBeanDefinitionReader的loadBeanDefinitions(resource)方法里面

1. 现在进入loadBeanDefinitions方法里面了, 目的很明确, 就是利用dom4j解析出xml中的id, class .scope

<bean id="petStore"

class="com.uknowzxt.service.v1.PetStoreService" >

</bean>

创建这个类的BeanDefinition对象, 并调用成员变量factory的registerBeanDefinition方法, 把类的BeanDefinition 放入factory的成员变量Map<String,BeanDefinition> beanDefinitionMap中.

顺便调用factory的setBeanClassLoader方法, 把classloader设置进去, getBean的时候要用

至此准备工作做完了, 要进行getBean了

三.涉及:

1.调用ApplicationContext的getBean方法, 因为factory中已经有了getBean()方法, 而且ApplicationContext也有factory的成员变量, 所以, 这里其实是直接调用factory的getBean方法.

2.现在进入getBean()方法了.

首先调用factory的getBeanDefinition(beanId)方法, 取出BeanDefinition. 有了BeanDefinition, 就可以生成实体类了

判断BeanDefinition的scope, 如果是非单例的直接进行类的创建.

如果是单例的, 在创建类之前, 先调用factory的getSingleton方法, 拿到单例类的Set集合中有没有已经创建好的类的实例. 如果没有, 进行类的创建. ---创建好了调用factory的registerSingleton(beanId,bean)方法, 把单例的类设置进去

1. 来研究一下createBean(BeanDefinition)方法, 通过beanDefinition拿到beanClassName, 利用类CLassLoader来获得类的Class对象, 利用反射来获得实例对象并返回

四: spring解耦做的很好, 每个功能几乎都做成了一个接口, 所以最后我们Factory的继承体系是这样的: ApplicationContext需要使用getBean, 和setBeanClassLoader方法, 所以它也在这个体系内:

