# 基础类:

## PropertyValue:

这是一个属性配置的包装类

private String name; //属性名

private Object value; //属性值

## PropertyValues

:这是一个对于属性的包装

private List<PropertyValue> list; //这是可以有多个不同的属性值

## BeanDefinition:

这是一个对象的包装类

private Object bean; //真正的对象

private Class beanClass; //真正的对象的Class

private String beanClassName; //真正的对象的class名称 比如:com.wanghao.test.User

private PropertyValues propertyValues; //上面的User的属性的集合 比如 int age; String name;

# 工厂类

## BeanFactory:

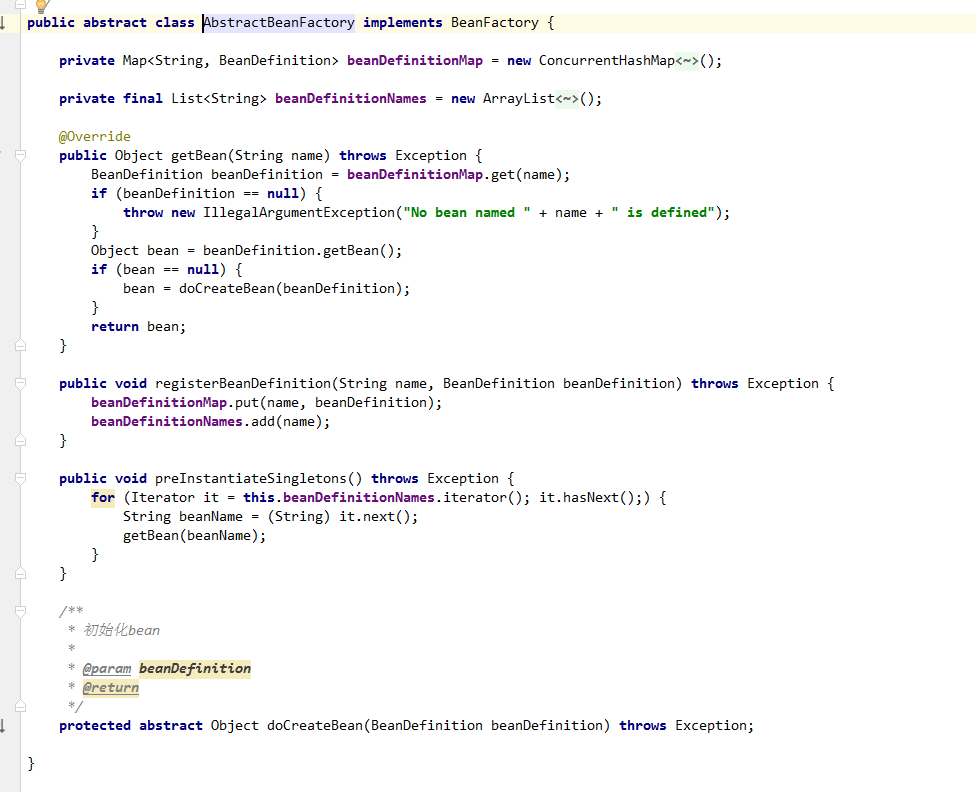
//获取对象

Object getBean(String name);

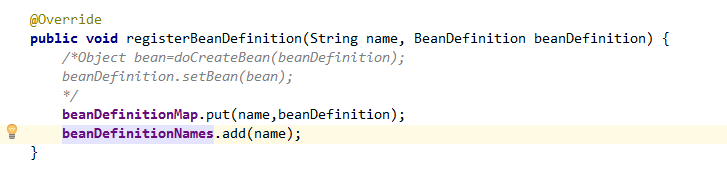
//注册对象  
**void** registerBeanDefinition(String name, BeanDefinition beanDefinition);

## **AbstractBeanFactory:**

Abstract class AbstractBeanFactory **implements** BeanFactory



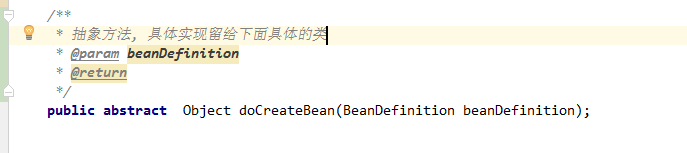
**对于BeanFactory里面的实现.先进行注册**



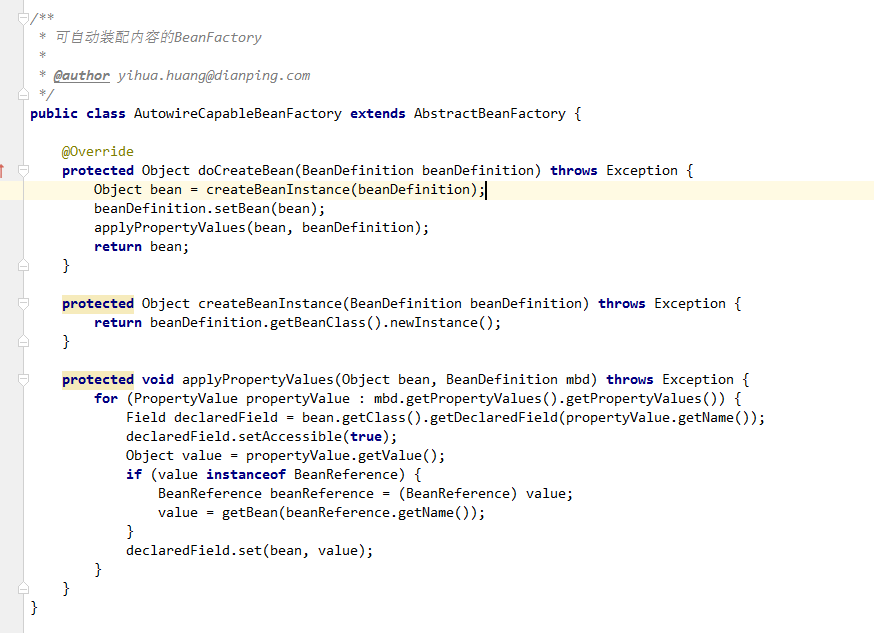
getBean() .先在beanDefinitionMap里面查看是否存在,不存在直接报错



//这里一个抽象类



## AutoWireCapableBeanFactory



把传递进来的BeanDefinition里面的bean设置上对象. 这叫对象的初始化.

真正的实现doCreateBean()创建对象

beanDefinition.getBeanClass().newInstance();这句话就是真的创建了对象.

beanDefinition.setBean(bean);//把对象设置到beanDefinition包装对象里面

applyPropertyValues(Object bean, BeanDefinition mbd)

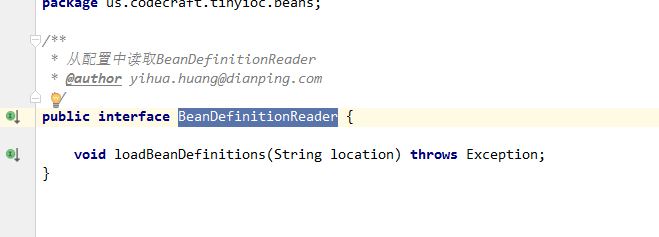
这个对于bean对象里面设置属性,遍历beanDefinition里面的propertyValues

,然后根据反射,Field.set方法, 设置到这个对象上面

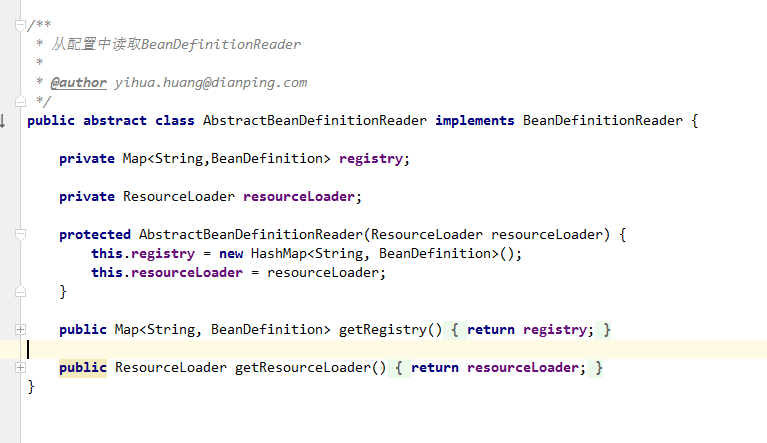
这样,整个对象就构建完成啦.

# Xml读取配置文件

## BeanDefinitionReader



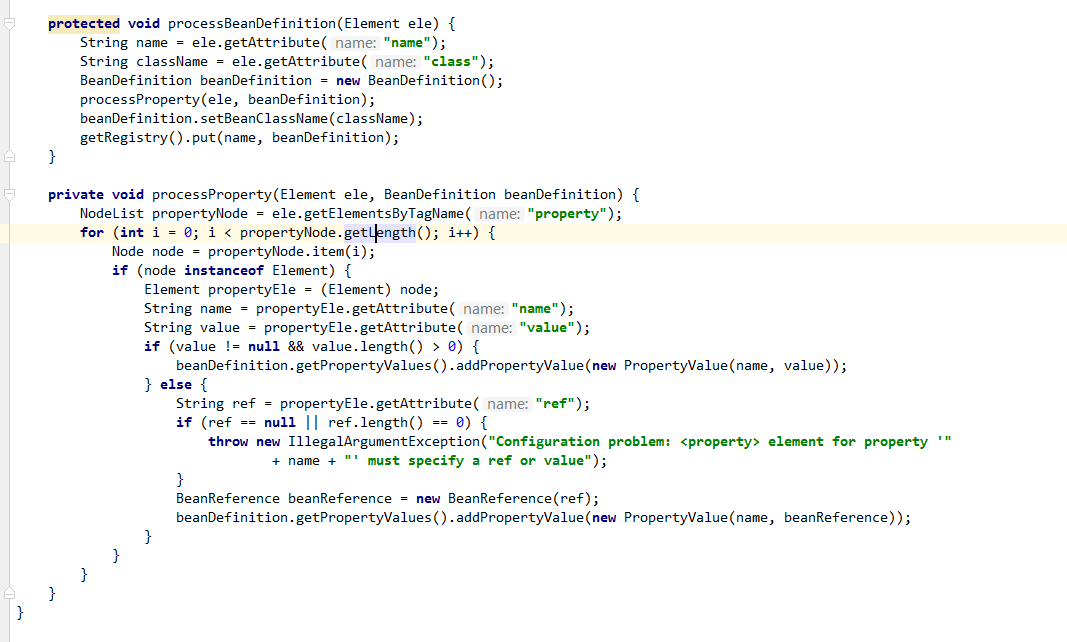
## AbstractBeanDefinitionReader:



## XMLBeanDefinitionReader

这是很厉害的一个类,很牛逼的哈





构造函数XmlBeanDefinitionReader需要一个ResourceLoader

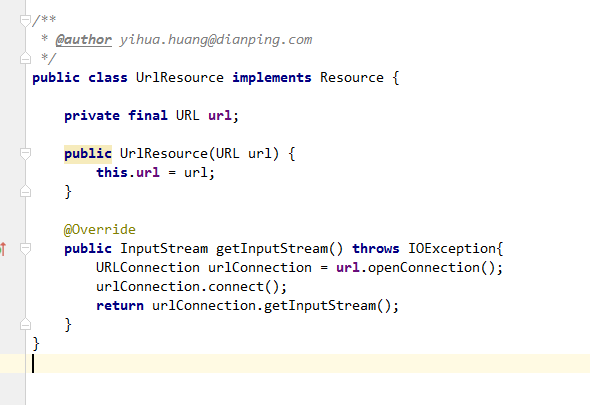
# IO获取spring-root.xml

## Resource

这是一个接口,提供了获取一个输入流的接口

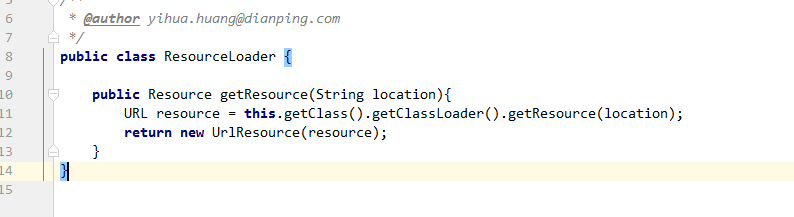


## URLResource



对于Resource的实现, 构造方法里面需要一个URL.统一资源定位,.比如(classPath:spring-root.xml). 在下面ResourceLoader用得到

## ResourceLoader



传递进去一个String的location位置,然后获取到一个Resource对象, 这个主要是通过classLoad来获取的.,然后把东西封装成URL对象. 最后可以获取inputStream啦.

ResourceLoader的作用就是返回一个Resource对象,可以获取inputStream流

# Context:上下文

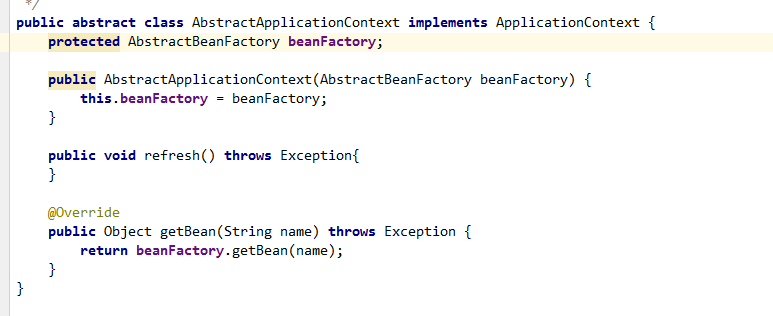
## ApplicationContext

ApplicationContext implement BeanFactory

但是里面什么都没有

## AbstractApplicationContext

AbstractApplicationContext implement ApplicationContext



抽象类里面有构造方法,在子类里面必须调用super方法 这个构造方法意思是说明,我这里面的BeanFactory需要被初始化,以便于后面用getBean的时候不会出现空指针异常问题.

## ClassPathXmlApplicationContext:

ClassPathXmlApplicationContext **extends** AbstractApplicationContext





对外提供的第一个构造方法,然后调用第二个构造方法

创建ClassPathXmlApplicationContext的时候就new了一个 AutowireCapableBeanFactory

这样BeanFactory有了对象

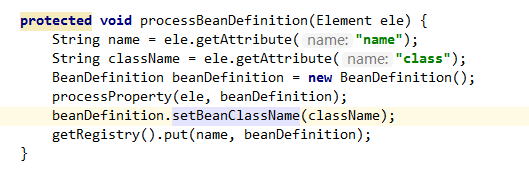
看下面这个refresh()方法.这个方法比较牛逼.

创建了一个XMLBeanDefinitionReader对象. 参数new ResourceLoader();

然后在xml配置文件里面new了一个Map<String,BeanDefinition>对象

然后调用loadBeanDefinitions(String config)//然后调用加载xml定义的方法

然后读取xml的就不说啦. 里面有

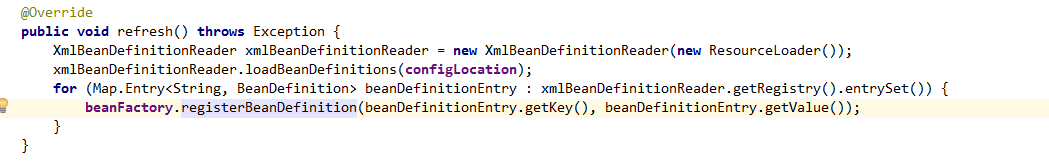


创建一个对象 BeanDefinition BeanDefinition=new BeanDefinition();

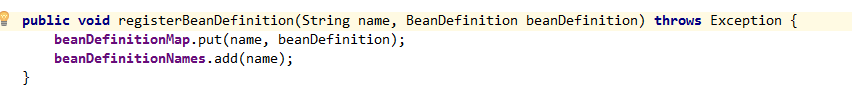
最后那句getRegistry().put(name, beanDefinition);

就是把BeanDefinition 设置到

**AbstractBeanDefinitionReader.registry 这个Map里面啦.**



下面这个for循环里面啦. 拿出啦里面刚才的内容,依次进行遍历循环



最后相当于放到了BeanFactory的里面 俩个属性里面.

# 重点:

**ClassPathXmlApplicationContext**, 读取配置xml文件,把配置文件里面的内容都转换成为BeanDefinition对象.然后放到BeanFactory里面的Map<String,BeanDefintion> 里面.

下面就是AbstractBeanFactory里面的getBean这个方法

第一步先在beanDefinitionMap里面获取BeanDefinition.然后再讲过这个包装类去获取里面的对象,如果对象为空, 然后调用创建对象的方法创建对象,

bean=doCreateBean(beanDefinition);

这里面有一个比较绕口的地方



Class HelloService{

Private OutService outservice;

}

Class OutService{

Private HelloService helloService;

}

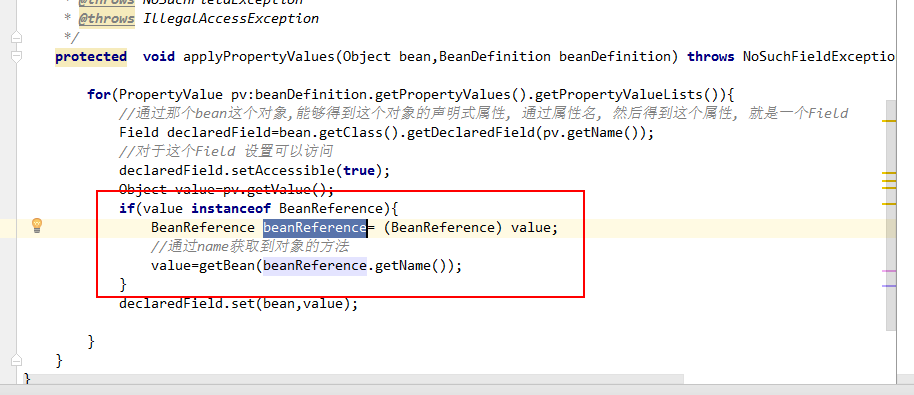
比如创建helloService

getBean(“helloService”)

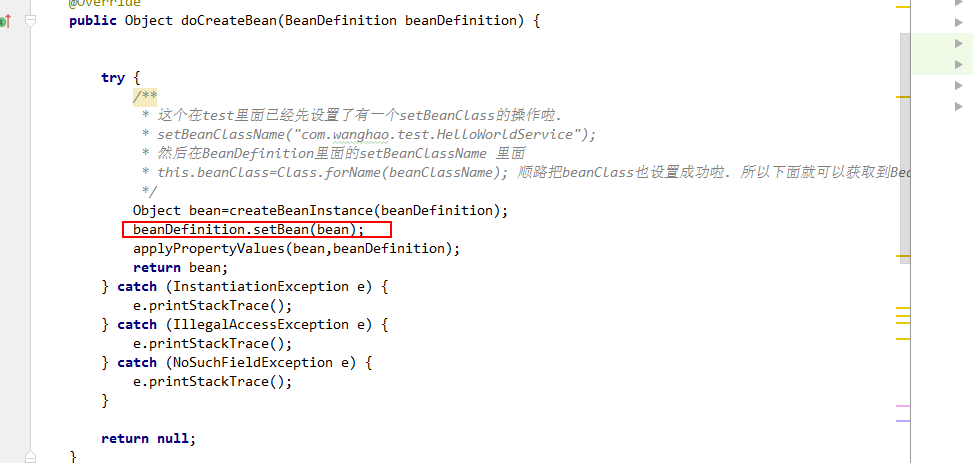
第一次再Map里面获取helloService的BeanDefinition.获取到了然后获取bean对象返回是空,执行doCreateBean方法, 通过反射得到Helloservice这个类的对象

**“BeanDefinition.setBean(bean)”**

**然后再执行属性注入,发现需要一个outService的引用.**



通过ref的名称去获取outService的对象,又走了getBean这个方法getBean一看在BeanDefinition.getBean 也是空, 也就是 doCreateBean



到了下面发现也是需要helloService这个对象,又去掉getBean

这次getBean里面有对象啦.



直接返回啦. 这样就把 outService装配完毕啦.

返回啦. 然后helloService也就能够装配完毕啦.

有点类似于 栈的概念.