

接上篇《Spring源码学习四：BeanDefinition装载前奏曲》，进入AbstractXmlApplicationContext类的loadBeanDefinitions方法，代码如下所示。

```
protected void loadBeanDefinitions(DefaultListableBeanFactory beanFactory) throws BeansException, IOException {
    // Create a new XmlBeanDefinitionReader for the given BeanFactory.
    XmlBeanDefinitionReader beanDefinitionReader = new
    XmlBeanDefinitionReader(beanFactory);

    // Configure the bean definition reader with this context's
    // resource loading environment.
    beanDefinitionReader.setEnvironment(this.getEnvironment());
    beanDefinitionReader.setResourceLoader(this);
    beanDefinitionReader.setEntityResolver(new ResourceEntityResolver(this));

    // Allow a subclass to provide custom initialization of the reader,
    // then proceed with actually loading the bean definitions.
    initBeanDefinitionReader(beanDefinitionReader);
    loadBeanDefinitions(beanDefinitionReader);
}
```

关注其中的最后一行的loadBeanDefinitions方法，具体实现也在AbstractXmlApplicationContext类中，代码如下所示。

```
protected void loadBeanDefinitions(XmlBeanDefinitionReader reader) throws BeansException, IOException {
    Resource[] configResources = getConfigResources();
    if (configResources != null) {
        reader.loadBeanDefinitions(configResources);
    }
    String[] configLocations = getConfigLocations();
    if (configLocations != null) {
        reader.loadBeanDefinitions(configLocations);
    }
}
```

这个方法里就是根据给定的xml配置文件进行后续的解析及装载。继续跟踪代码，进入reader.loadBeanDefinitions(configLocations)，这个XmlBeanDefinitionReader用于从xml文件中读取Bean的定义。接下来的代码有很多跳转，大部分都是不重要的，直接追踪到重要代码，代码如下。

```
public int loadBeanDefinitions(EncodedResource encodedResource) throws BeanDefinitionStoreException {
    Assert.notNull(encodedResource, "EncodedResource must not be null");
    if (logger.isInfoEnabled()) {
        logger.info("Loading XML bean definitions from " + encodedResource.getResource());
    }

    Set<EncodedResource> currentResources = this.resourcesCurrentlyBeingLoaded.get();
    if (currentResources == null) {
        currentResources = new HashSet<EncodedResource>(4);
        this.resourcesCurrentlyBeingLoaded.set(currentResources);
    }
    if (!currentResources.add(encodedResource)) {
        throw new BeanDefinitionStoreException(
            "Detected cyclic loading of " + encodedResource + " - check your import definitions!");
    }
    try {
        InputStream inputStream = encodedResource.getResource().getInputStream();
        try {
            InputSource inputSource = new InputSource(inputStream);
            if (encodedResource.getEncoding() != null) {
                inputSource.setEncoding(encodedResource.getEncoding());
            }
            return doLoadBeanDefinitions(inputSource, encodedResource.getResource());
        }finally {
            inputStream.close();
        }
    }catch (IOException ex) {
        throw new BeanDefinitionStoreException(
            "IOException parsing XML document from " + encodedResource.getResource(), ex);
    }finally {
        currentResources.remove(encodedResource);
        if (currentResources.isEmpty()) {
            this.resourcesCurrentlyBeingLoaded.remove();
        }
    }
}
```

这里只要关注doLoadBeanDefinitions方法，进入其实现，代码如下。

```
protected int doLoadBeanDefinitions(InputSource inputSource, Resource resource)
    throws BeanDefinitionStoreException {
    try {
        Document doc = doLoadDocument(inputSource, resource);
        return registerBeanDefinitions(doc, resource);
    }catch (BeanDefinitionStoreException ex) {
        throw ex;
    }catch (SAXParseException ex) {
        throw new XmlBeanDefinitionStoreException(resource.getDescription(),
            "Line " + ex.getLineNumber() + " in XML document from " + resource + " is invalid", ex);
    }catch (SAXException ex) {
        throw new XmlBeanDefinitionStoreException(resource.getDescription(),
            "XML document from " + resource + " is invalid", ex);
    }catch (ParserConfigurationException ex) {
        throw new BeanDefinitionStoreException(resource.getDescription(),
            "Parser configuration exception parsing XML from " + resource, ex);
    }catch (IOException ex) {
        throw new BeanDefinitionStoreException(resource.getDescription(),
            "IOException parsing XML document from " + resource, ex);
    }catch (Throwable ex) {
        throw new BeanDefinitionStoreException(resource.getDescription(),
            "Unexpected exception parsing XML document from " + resource, ex);
    }
}
```

直接看try块中的代码，可以看到，Spring将xml配置文件转成Document，这里使用了SAX对XML的解析，至于里面是如何解析可以后续针对性的研究。接着进入registerBeanDefinitions方法，后续又有很多代码的跳转，先不——关注，直接进入重要代码，代码如下。

```
protected void parseBeanDefinitions(Element root, BeanDefinitionParserDelegate delegate) {
    if (delegate.isDefaultNamespace(root)) {
        NodeList nl = root.getChildNodes();
        for (int i = 0; i < nl.getLength(); i++) {
            Node node = nl.item(i);
            if (node instanceof Element) {
                Element ele = (Element) node;
                if (delegate.isDefaultNamespace(ele)) {
                    parseDefaultElement(ele, delegate);
                }else {
                    delegate.parseCustomElement(ele);
                }
            }
        }
    }else {
        delegate.parseCustomElement(root);
    }
}
```

可以看到就是对Document中元素、节点的不断解析。这里的解析分成了两条路线，一个是默认标签的解析，如Spring自己定义的标签；一个是对自定义标签的解析，如自定义的标签。这里先关注默认标签的解析，对于自定义标签的解析可以后续分析，进入parseDefaultElement方法，代码如下。

```
private void parseDefaultElement(Element ele, BeanDefinitionParserDelegate delegate) {
    if (delegate.nodeNameEquals(ele, IMPORT_ELEMENT)) {
        importBeanDefinitionResource(ele);
    }else if (delegate.nodeNameEquals(ele, ALIAS_ELEMENT)) {
        processAliasRegistration(ele);
    }else if (delegate.nodeNameEquals(ele, BEAN_ELEMENT)) {
        processBeanDefinition(ele, delegate);
    }else if (delegate.nodeNameEquals(ele, NESTED_BEANS_ELEMENT)) {
        // recurse
        doRegisterBeanDefinitions(ele);
    }
}
```

代码中分别对import、alias、bean、beans标签进行解析，最终将解析得到的BeanDefinition存入DefaultListableBeanFactory中的beanDefinitionMap中。