

12-07 Spring源码编译和xml解析

1、spring为什么学

1. 其他框架用到了，比如：如果使用netty,会使用到spring
2. 代码更加优雅，体现程序员功力，不喜欢屎山
3. 面试需要

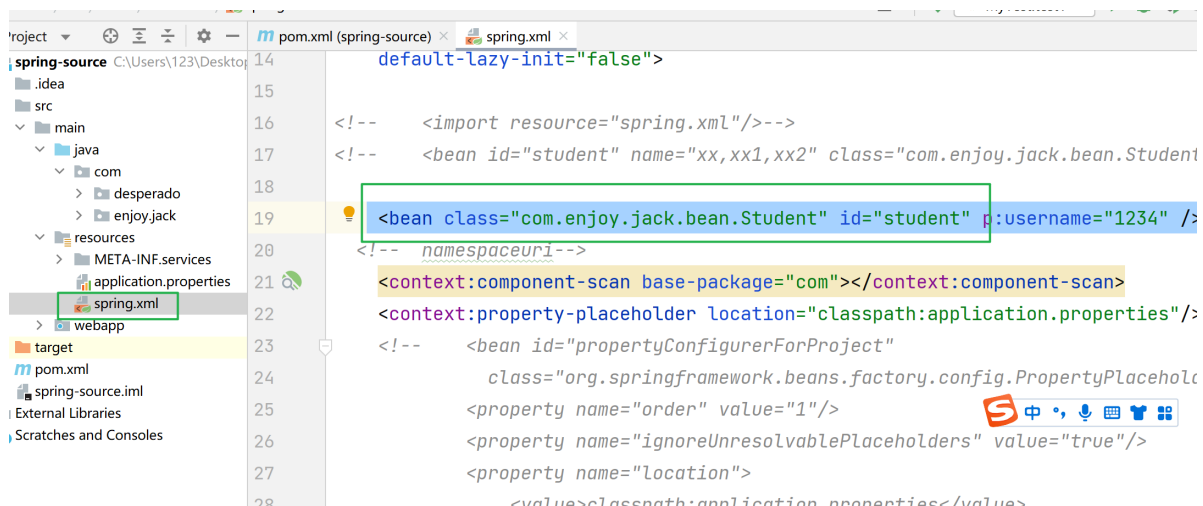
2、编译-代码

spring源码工程中也用到了 命令 bat

<spring.version>5.2.8.RELEASE</spring.version> -----源码版本

3、代码分析

[http://117.33.237.52:20070/desperado-image/xml%E8%A7%A3%E6%9E%90%E5%92%8CBeanDefinition%E5%B0%81%E8%A3%85%E6%A0%B8%E5%BF%83%E6%96%B9%E6%B3%95%20refreshBeanFactory\(\).jpg](http://117.33.237.52:20070/desperado-image/xml%E8%A7%A3%E6%9E%90%E5%92%8CBeanDefinition%E5%B0%81%E8%A3%85%E6%A0%B8%E5%BF%83%E6%96%B9%E6%B3%95%20refreshBeanFactory().jpg)



```
public void testDcard(){
    ClassPathXmlApplicationContext application = new ClassPathXmlApplicationContext( configLocation:
    //application.addApplicationListener(new Event("event", "jack"));
    //application.publishEvent(new Event("Jack", "enjoyEvent"));

    Student bean = application.getBean(Student.class);
    System.out.println(bean);
}
```

```
public class Student {
    private String username = "jack";
}
```

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${spring.version}</version>
</dependency>
```

模板设计模式

钩子方法。com.enjoy.jack.designPattern.template

```

@Override
public void postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory) throws BeansException {
    DefaultListableBeanFactory beanFactory1 = (DefaultListableBeanFactory) beanFactory;
    beanFactory1.setAllowBeanDefinitionOverriding(true);
    beanFactory1.setAllowCircularReferences(true);
}

@Component
public class BeanProcessor implements BeanDefinitionRegistryPostProcessor {

    @Override
    public void postProcessBeanDefinitionRegistry(BeanDefinitionRegistry registry) throws BeansException {
        beanFactory.setSerializationId(getId());
        // 设置是否可以循环依赖 allowCircularReferences
        // 是否允许使用相同名称重新注册不同的bean实现。
        customizeBeanFactory(beanFactory);
        // 解析xml，并把xml中的标签封装成BeanDefinition对象
        loadBeanDefinitions(beanFactory);
    }
}

```

委托设计模式

com.enjoy/jack/designPattern/entrust

```

public int loadBeanDefinitions(String location, @Nullable Set<Resource> actualResources) throws BeansException {
    ResourceLoader resourceLoader = getResourceLoader();
    if (resourceLoader == null) {
        throw new BeansException("Cannot load bean definitions: no ResourceLoader available");
    }
    if (resourceLoader instanceof ResourcePatternResolver) {
        ResourcePatternResolver resourceLoader1 = (ResourcePatternResolver) resourceLoader;
        if (resourceLoader1.isInstance(resourcePatternResolver)) {
            // Resource pattern resolver available
        }
    }
}

```

Result:

```

00 result = (ClassPathXmlApplicationContext@1309) ... V
01 configResources = null
02 resourceLoader = org.springframework.core.io.support.PathMatchingResourcePatternResolver
03 resourcePatternResolver = org.springframework.core.io.support.PathMatchingResourcePatternResolver
04 resourcePatternResolver.isInstance(resourcePatternResolver) = true
05 resourcePatternResolver.isInstance(resourcePatternResolver) = true
06 resourcePatternResolver.isInstance(resourcePatternResolver) = true
07 resourcePatternResolver.isInstance(resourcePatternResolver) = true
08 resourcePatternResolver.isInstance(resourcePatternResolver) = true
09 resourcePatternResolver.isInstance(resourcePatternResolver) = true
10 resourcePatternResolver.isInstance(resourcePatternResolver) = true
11 resourcePatternResolver.isInstance(resourcePatternResolver) = true
12 resourcePatternResolver.isInstance(resourcePatternResolver) = true
13 resourcePatternResolver.isInstance(resourcePatternResolver) = true
14 resourcePatternResolver.isInstance(resourcePatternResolver) = true
15 resourcePatternResolver.isInstance(resourcePatternResolver) = true
16 resourcePatternResolver.isInstance(resourcePatternResolver) = true
17 resourcePatternResolver.isInstance(resourcePatternResolver) = true
18 resourcePatternResolver.isInstance(resourcePatternResolver) = true
19 resourcePatternResolver.isInstance(resourcePatternResolver) = true
20 resourcePatternResolver.isInstance(resourcePatternResolver) = true
21 resourcePatternResolver.isInstance(resourcePatternResolver) = true
22 resourcePatternResolver.isInstance(resourcePatternResolver) = true
23 resourcePatternResolver.isInstance(resourcePatternResolver) = true
24 resourcePatternResolver.isInstance(resourcePatternResolver) = true
25 resourcePatternResolver.isInstance(resourcePatternResolver) = true
26 resourcePatternResolver.isInstance(resourcePatternResolver) = true
27 resourcePatternResolver.isInstance(resourcePatternResolver) = true
28 resourcePatternResolver.isInstance(resourcePatternResolver) = true
29 resourcePatternResolver.isInstance(resourcePatternResolver) = true
30 resourcePatternResolver.isInstance(resourcePatternResolver) = true
31 resourcePatternResolver.isInstance(resourcePatternResolver) = true
32 resourcePatternResolver.isInstance(resourcePatternResolver) = true
33 resourcePatternResolver.isInstance(resourcePatternResolver) = true
34 resourcePatternResolver.isInstance(resourcePatternResolver) = true
35 resourcePatternResolver.isInstance(resourcePatternResolver) = true
36 resourcePatternResolver.isInstance(resourcePatternResolver) = true
37 resourcePatternResolver.isInstance(resourcePatternResolver) = true
38 resourcePatternResolver.isInstance(resourcePatternResolver) = true
39 resourcePatternResolver.isInstance(resourcePatternResolver) = true
40 resourcePatternResolver.isInstance(resourcePatternResolver) = true
41 resourcePatternResolver.isInstance(resourcePatternResolver) = true
42 resourcePatternResolver.isInstance(resourcePatternResolver) = true
43 resourcePatternResolver.isInstance(resourcePatternResolver) = true
44 resourcePatternResolver.isInstance(resourcePatternResolver) = true
45 resourcePatternResolver.isInstance(resourcePatternResolver) = true
46 resourcePatternResolver.isInstance(resourcePatternResolver) = true
47 resourcePatternResolver.isInstance(resourcePatternResolver) = true
48 resourcePatternResolver.isInstance(resourcePatternResolver) = true
49 resourcePatternResolver.isInstance(resourcePatternResolver) = true
50 resourcePatternResolver.isInstance(resourcePatternResolver) = true
51 resourcePatternResolver.isInstance(resourcePatternResolver) = true
52 resourcePatternResolver.isInstance(resourcePatternResolver) = true
53 resourcePatternResolver.isInstance(resourcePatternResolver) = true
54 resourcePatternResolver.isInstance(resourcePatternResolver) = true
55 resourcePatternResolver.isInstance(resourcePatternResolver) = true
56 resourcePatternResolver.isInstance(resourcePatternResolver) = true
57 resourcePatternResolver.isInstance(resourcePatternResolver) = true
58 resourcePatternResolver.isInstance(resourcePatternResolver) = true
59 resourcePatternResolver.isInstance(resourcePatternResolver) = true
60 resourcePatternResolver.isInstance(resourcePatternResolver) = true
61 resourcePatternResolver.isInstance(resourcePatternResolver) = true
62 resourcePatternResolver.isInstance(resourcePatternResolver) = true
63 resourcePatternResolver.isInstance(resourcePatternResolver) = true
64 resourcePatternResolver.isInstance(resourcePatternResolver) = true
65 resourcePatternResolver.isInstance(resourcePatternResolver) = true
66 resourcePatternResolver.isInstance(resourcePatternResolver) = true
67 resourcePatternResolver.isInstance(resourcePatternResolver) = true
68 resourcePatternResolver.isInstance(resourcePatternResolver) = true
69 resourcePatternResolver.isInstance(resourcePatternResolver) = true
70 resourcePatternResolver.isInstance(resourcePatternResolver) = true
71 resourcePatternResolver.isInstance(resourcePatternResolver) = true
72 resourcePatternResolver.isInstance(resourcePatternResolver) = true
73 resourcePatternResolver.isInstance(resourcePatternResolver) = true
74 resourcePatternResolver.isInstance(resourcePatternResolver) = true
75 resourcePatternResolver.isInstance(resourcePatternResolver) = true
76 resourcePatternResolver.isInstance(resourcePatternResolver) = true
77 resourcePatternResolver.isInstance(resourcePatternResolver) = true
78 resourcePatternResolver.isInstance(resourcePatternResolver) = true
79 resourcePatternResolver.isInstance(resourcePatternResolver) = true
80 resourcePatternResolver.isInstance(resourcePatternResolver) = true
81 resourcePatternResolver.isInstance(resourcePatternResolver) = true
82 resourcePatternResolver.isInstance(resourcePatternResolver) = true
83 resourcePatternResolver.isInstance(resourcePatternResolver) = true
84 resourcePatternResolver.isInstance(resourcePatternResolver) = true
85 resourcePatternResolver.isInstance(resourcePatternResolver) = true
86 resourcePatternResolver.isInstance(resourcePatternResolver) = true
87 resourcePatternResolver.isInstance(resourcePatternResolver) = true
88 resourcePatternResolver.isInstance(resourcePatternResolver) = true
89 resourcePatternResolver.isInstance(resourcePatternResolver) = true
90 resourcePatternResolver.isInstance(resourcePatternResolver) = true
91 resourcePatternResolver.isInstance(resourcePatternResolver) = true
92 resourcePatternResolver.isInstance(resourcePatternResolver) = true
93 resourcePatternResolver.isInstance(resourcePatternResolver) = true
94 resourcePatternResolver.isInstance(resourcePatternResolver) = true
95 resourcePatternResolver.isInstance(resourcePatternResolver) = true
96 resourcePatternResolver.isInstance(resourcePatternResolver) = true
97 resourcePatternResolver.isInstance(resourcePatternResolver) = true
98 resourcePatternResolver.isInstance(resourcePatternResolver) = true
99 resourcePatternResolver.isInstance(resourcePatternResolver) = true

```

org.springframework.context.support.AbstractXmlApplicationContext

loadBeanDefinitions

MyTest.testCan

AbstractBeanDefinitionReader.java

AbstractXmlApplicationContext.java

```

80 @Override
81 protected void loadBeanDefinitions(DefaultListableBeanFactory beanFactory) throws BeansException {
82     // Create a new XmlBeanDefinitionReader for the given BeanFactory.
83     // 创建xml的解析器，这里是一个委托模式
84     XmlBeanDefinitionReader beanDefinitionReader = new XmlBeanDefinitionReader(beanFactory);
85
86     // Configure the bean definition reader with this context's
87     // resource loading environment.
88     beanDefinitionReader.setEnvironment(this.getEnvironment());
89     // 这里传一个this进去，因为ApplicationContext是实现了ResourceLoader接口的
90     beanDefinitionReader.setResourceLoader(this);
91     beanDefinitionReader.setEntityResolver(new DefaultEntityResolver());
92
93     // Allow a subclass to provide custom initialization
94     // then proceed with actually loading the bean definitions
95     initBeanDefinitionReader(beanDefinitionReader);
96     // 主要看这个方法 重要程度 5
97     loadBeanDefinitions(beanDefinitionReader);
98 }
99

```

public void setResourceLoader(@Nullable org.springframework.core.io.ResourceLoader resourceLoader) {

Set the ResourceLoader to use for resource locations. If ResourcePatternResolver, the bean definition reader will use resource patterns to Resource arrays. Default is PathMatchingResourcePatternResolver, also pattern resolving through the ResourcePatternResolver. Setting this to null suggests that absolute resource locations should be used by this bean definition reader.

See Also: ResourcePatternResolver, PathMatchingResourcePatternResolver

org.springframework.beans.factory.support.AbstractBeanDefinitionReader

上下文接口实现资源加载接口

```

12 // Resource pattern matching available.
13 try {
14     // 把字符串类型的xml文件路径，形如：classpath*:user/**/*-context.xml，转换成Resource对象类型，其实
15     // 用流
16     // 的方式加载配置文件，然后封装成Resource对象，不重要，可以不看
17     Resource[] resources = ((ResourcePatternResolver) resourceLoader).getResources(location);
18     // 主要看这个方法 ** 重要程度 5
19     int count = loadBeanDefinitions(resources);
20     if (actualResources != null) {
21         Collections.addAll(actualResources, resources);
22     }

```

//EncodedResource 带编码的对Resource对象的封装

```
return loadBeanDefinitions(new EncodedResource(resource));
```

```
try {
    InputStream inputStream = encodedResource.getResource().getInputStream();

```

//InputStream是jdk中的sax xml文件解析对象

```
InputStream inputStream = new InputStream(inputStream);
```

```
if (encodedResource.getEncoding() != null) {
```

```
    inputStream.setEncoding(encodedResource.getEncoding());
```

```
}
```

//主要看这个方法 ** 重要程度 5

```
return doLoadBeanDefinitions(inputStream, encodedResource.getResource());
```

```
try {
```

//把inputSource 封装成Document文件对象，这是jdk的API

```
Document doc = doLoadDocument(inputStream, resource);
```

//主要看这个方法，根据解析出来的document对象，拿到里面的标签元素封装成BeanDefinition

```
int count = registerBeanDefinitions(doc, resource);
```

//import标签解析 重要程度 1，可看可不看

```
if (delegate.nodeNameEquals(ele, IMPORT_ELEMENT)) {
```

```
    importBeanDefinitionResource(ele);
```

```
}
```

//alias标签解析 别名标签 重要程度 1，可看可不看

```
else if (delegate.nodeNameEquals(ele, ALIAS_ELEMENT)) {
```

```
    processAliasRegistration(ele);
```

```
}
```

//bean标签，重要程度 5，必须看

```
else if (delegate.nodeNameEquals(ele, BEAN_ELEMENT)) {
```

```
    processBeanDefinition(ele, delegate);
```

```
}
```

```
else if (delegate.nodeNameEquals(ele, NESTED_BEANS_ELEMENT)) {
```

// recurse

```
doRegisterBeanDefinitions(ele);
```

```
}
```

```
}
```

<http://117.33.237.52:20070/desperado-image/GenericBeanDefinition.jpg>