 学习过Spring的同学都知道，Spring框架的核心就是IoC和AOP。Spring可以**理解为一个工厂，负责对象的创建和对象间关系的维护**。IoC即控制反转，简单点说就是原来的对象是在要使用之前通过在代码里通过new 的方式创建出来的而IOC的思想则是由**spring容器创建统一创建(配置文件中注册bean对象)，**在程序要使用到该对象的时候，**自动注入**。（spring默认在web容器启动的时候就**创建了单例的对象**）其最大的作用是减少了代码之间的耦合度。

     下面是我自己写的一个Spring框架，只是简单的模拟和实现了Spring的IoC容器功能。

思路如下：

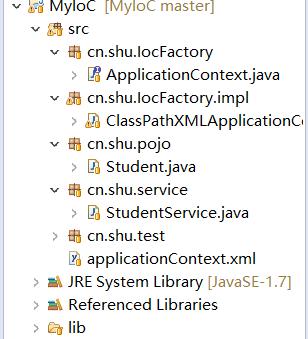
1. **解析xml配置文件**
2. **根据配置的生成相应的对象**
3. **将对象存入IOC容器**

步骤如下：

1. 创建bean类和对应的Service类
2. 配置xml解析文件（Spring的配置文件：application.xml.）
3. 最后也是实现Ioc最为核心的部分ClassPathXMLApplicationContext。这个类采用Jdom对application.xml文档进行解析。

大致思路是：通过**读取配置文件中定义的全路径类名，利用java反射技术创建该类的实例。**首先定义HashMap这样的一重数据结构用于存储创建的实例。Key值就是配置文件的<bean>标签“id”,value值就是对应的实例。然后该类需要提供  public Object getBean(String name)方法返回实例对象。

 创建一个java/web工程：MyIoC .  目录结构如下：



下面对上面所创建的类逐一介绍。

        首先创建一个Student类和StudentService类如下：

**[java]** [view plain](http://blog.csdn.net/u010994169/article/details/71169986) [copy](http://blog.csdn.net/u010994169/article/details/71169986)

1. **package** cn.shu.pojo;
2. **public** **class** Student {
4. **private** String name;
5. **private** String add;
7. **public** **void** selfIntroDuction(){
8. System.out.println("我的姓名是 " + name + " 我来自 " + add);
10. }
12. **public** String getName() {
13. **return** name;
14. }
16. **public** **void** setName(String name) {
17. **this**.name = name;
18. }
20. **public** String getAdd() {
21. **return** add;
22. }
24. **public** **void** setAdd(String add) {
25. **this**.add = add;
26. }
28. }

**[java]** [view plain](http://blog.csdn.net/u010994169/article/details/71169986) [copy](http://blog.csdn.net/u010994169/article/details/71169986)

1. <pre name="code" **class**="java">**package** cn.shu.service;
2. **import** cn.shu.pojo.Student;
4. **public** **class** StudentService {
5. **private**  Student student;
7. **public** Student getStudent() {
8. **return** student;
9. }
11. **public** **void** setStudent(Student student) {
12. **this**.student = student;
13. }
14. }

         接着编写Spring的配置文件：application.xml.

**[html]** [view plain](http://blog.csdn.net/u010994169/article/details/71169986) [copy](http://blog.csdn.net/u010994169/article/details/71169986)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
3. **<beans>**
4. **<bean** id="Student" class="cn.shu.pojo.Student"**>**
5. **<property** name="name" value="Hover"**/>**
6. **<property** name="add" value="China"**/>**
7. **</bean>**
9. **<bean** id="StudentService" class="cn.shu.service.StudentService"**>**
10. **<property** ref="Student"**/>**
11. **</bean>**
12. **</beans>**

     最后也是实现Ioc最为核心的部分ClassPathXMLApplicationContext。这个类采用Jdom对application.xml文档进行解析。大致思路是：**通过读取配置文件中定义的全路径类名，利用java反射技术创建该类的实例**。首先定义HashMap这样的一重数据结构用于存储创建的实例。Key值就是配置文件的<bean>标签“id”,value值就是对应的实例。然后该类需要提供  public Object getBean(String name)方法返回实例对象。具体代码如下：

**[java]** [view plain](http://blog.csdn.net/u010994169/article/details/71169986) [copy](http://blog.csdn.net/u010994169/article/details/71169986)

1. **package** cn.shu.IocFactory.impl;
3. **import** org.jdom.input.SAXBuilder;
4. **import** org.jdom.Document;
5. **import** org.jdom.JDOMException;
6. **import** org.jdom.Element;
7. **import** org.jdom.xpath.XPath;
9. **import** java.io.File;
10. **import** java.io.IOException;
11. **import** java.net.URL;
12. **import** java.net.URISyntaxException;
13. **import** java.util.\*;
14. **import** java.lang.reflect.Field;
15. **import** java.lang.reflect.Method;
16. **import** java.lang.reflect.InvocationTargetException;
18. **import** cn.shu.IocFactory.ApplicationContext;
20. **public** **class** ClassPathXMLApplicationContext **implements** ApplicationContext{

23. **private** File file;
24. **private** Map map = **new** HashMap();
26. **public** ClassPathXMLApplicationContext(String config\_file) {
27. URL url = **this**.getClass().getClassLoader().getResource(config\_file);
29. **try** {
30. file = **new** File(url.toURI());
31. XMLParsing();
32. } **catch** (Exception e) {
33. // TODO Auto-generated catch block
34. e.printStackTrace();
35. }
37. }
39. **private** **void** XMLParsing() **throws** Exception {
40. SAXBuilder builder = **new** SAXBuilder();
41. Document doc = builder.build(file);
42. XPath xpath = XPath.newInstance("//bean");
43. List beans = xpath.selectNodes(doc);
44. Iterator i = beans.iterator();
45. **while** (i.hasNext()) {  **//循环获取bean里所有配置的类**
46. Element bean = (Element) i.next();  **//读取bean配置文件**
47. String id = bean.getAttributeValue("id");  **//读取文件路径**
48. String cls = bean.getAttributeValue("class");
49. Object obj = Class.forName(cls).newInstance();  **//反射创建对象**
50. Method[] method = obj.getClass().getDeclaredMethods();/**/获取方法**
51. List<Element> list = bean.getChildren("property"); /**/获取方法,把方法set进bean里，让你实现动态调用**
52. **for** (Element el : list) {
53. **for** (**int** n = 0; n < method.length; n++) {
54. String name = method[n].getName();
55. String temp = **null**;
56. **if** (name.startsWith("set")) {
57. temp = name.substring(3, name.length()).toLowerCase();
58. **if** (el.getAttribute("name") != **null**) {
59. **if** (temp.equals(el.getAttribute("name").getValue())) {
60. method[n].invoke(obj, el.getAttribute("value").getValue());
61. }
62. } **else** {
63. method[n].invoke(obj,map.get(el.getAttribute("ref").getValue()));
64. }
65. }
66. }
67. }
68. map.put(id, obj); **//用HashMap存储实例。id为key,如studentService。**
69. }
70. }
72. **public** Object getBean(String name) {
73. **return** map.get(name);  /**/取值，如取studentService**。
74. }
76. }

  编写测试类：

**[java]** [view plain](http://blog.csdn.net/u010994169/article/details/71169986) [copy](http://blog.csdn.net/u010994169/article/details/71169986)

1. **package** cn.shu.test;
3. **import** cn.shu.IocFactory.ApplicationContext;
4. **import** cn.shu.IocFactory.impl.ClassPathXMLApplicationContext;
5. **import** cn.shu.service.StudentService;
7. **public** **class** Test {
8. **public** **static** **void** main(String[] args) {
9. ApplicationContext context = **new** ClassPathXMLApplicationContext("applicationContext.xml");
10. StudentService  stuServ = (StudentService) context.getBean("StudentService");
11. stuServ.getStudent().selfIntroDuction();
12. }
14. }

测试结果如下：

