其实这边就是对前面研究过的spring管理bean的扩充，容我慢慢细讲

一样的配方我就说了，我来罗列一下不一样的配方

先看bean

```
public class Bean {

    private String id;
    private String className;
    private List<Property> propertyList;
```

多了一个List属性

这个Property是新的持久化类

```
public class Property {

    private String name;
    private String ref;
```

再看beanShow的构造方法

```
public beanShow(String xml) {
    readXML(xml);
    instanceBeans();
    injectObject();
}
```

可以看到多初始化了一个方法，而这个方法就是用来为bean对象的属性注入值

我们看看reaadXml的变化

```
for (Element element : list) {
    String id = element.getAttributeValue("id");
    String className = element.getAttributeValue("class");
    // 获取每个bean下面的属性
    List<Element> listt = element.getChildren("property", xhtml);
    List<Property> plist = new ArrayList<Property>();
    if (listt.size() > 0) {
        for (Element elementt : listt) {
            String name = elementt.getAttributeValue("name");
            String ref = elementt.getAttributeValue("ref");
            Property property = new Property();
            property.setName(name);
            property.setRef(ref);
            plist.add(property);// 保存属性节点
        }
    }
    Bean bean = new Bean();
    bean.setId(id);
    bean.setClassName(className);
    bean.setPropertyList(plist);
    beanList.add(bean);
}
```

就从这个for循环开始看，因为xml文件中用了依赖注入，bean不是有了个子标签property吗，所以这边也得再往下遍历判断一次

我们再来看看我们的主角injectObject

```
public void injectObject() {
    for (Bean bean : beanList) {
        Object object = beanObject.get(bean.getId());
        if (object != null) {
            try {
                PropertyDescriptor[] ps = Introspector.getBeanInfo(
                        object.getClass()).getPropertyDescriptors(); // 取得bean的属性描述
                for (Property property : bean.getPropertyList()) // 获取bean节点的属性
                {
                    for (PropertyDescriptor properdesc : ps) {
                        if (property.getName().equals(properdesc.getName())) {
                            Method setter = properdesc.getWriteMethod();// 获取属性的setter方法
                                                                        // ,private
                            if (setter != null) {
                                Object value = beanObject.get(property.getRef()); // 取得值
                                setter.setAccessible(true); // 设置为允许访问
                                setter.invoke(object, value);// 把引用对象注入到属性
                            }
                            break;
                        }
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

如之前的描述，它就是用来bean对象的属性注入值的

运行测试方法

```
信息: Pre-1
你好，鲁鲁
```