说实话，这也是我耐着性子跟着做完的，本身是对研究这种原理不是太感兴趣的，但是区别于简单地敲代码好像就是去研究这些原理，而且对自己使用有帮助（手动笑哭）

进入整题

其实这个原理理清楚了就是如何去解析xml里面的bean

bean的构造就是一个id加一个class

首先我们定义一个bean的持久化类

```
public class Bean {

    private String id;
    private String className;
```

两个属性对应起来

然后就是我们主体的操作类

```
public class beanShow {


    private List<Bean> beanList = new ArrayList<Bean>();
    private Map beanObject =new HashMap();

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

list就是将获取的bean的id和class记录下来

map则是真正落实让相应的bean起作用，就是通过这些bean管理持久化类

构造方法很明了，这两个方法下面也会介绍

读取xml配置文件

```
 private void readXML(String filename) {
     // 寻找配置文件
     URL xmlPath = this.getClass().getClassLoader().getResource(filename);
     Document doc = null;
     Element root = null;
     try {
         //使用JDOM首先要指定使用什么解析器,
         SAXBuilder sb = new SAXBuilder(false);//表示使用默认的解析器
         doc = sb.build(new FileInputStream(new File(xmlPath.toURI())));
         // 设置命名空间
         Namespace xhtml = Namespace.getNamespace("xhtml", "http://www.springframework.org/schema/
         root=doc.getRootElement();//获取根元素
         List<Element> list=root.getChildren("bean",xhtml);
         for(Element element : list){
             String id=element.getAttributeValue("id");
             String className=element.getAttributeValue("class");
             Bean bean=new Bean();
             bean.setId(id);
             bean.setClassName(className);
             beanList.add(bean);
         }

     } catch (Exception e) {
         e.printStackTrace();
     }
 }
```

看到一些奇怪的类不要慌，这里其实导入了一个jar包jdom.jar，这些类都是这里面的，jdom其实就是一个xml解析工具，如图所示，它做的也正是这个工作

bean的实例化

```
private void instanceBeans(){
    for(Bean bean : beanList){
        try{
            if (bean.getClassName() != null && !"".equals(bean.getClassName().trim()))
                beanObject.put(bean.getId(), Class.forName(bean.getClassName()).newInstance());
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

获取bean实例，在对象实例化时使用，原本的也是有的

```
public Object getBean(String beanName){
    return this.beanObject.get(beanName);
}
```

可以看到这边已经在用我们自己解析器读取xml文件了

```
@Test
public void testBean() throws Exception{
    beanShow bs = new beanShow("applicationContext.xml");
    Person p=(Person)bs.getBean("userBean");
    p.show();
}
```

```
hello kugou
```

结果也是没有没有问题的

把封装好的xml解析工作明面化，其实也就是这么回事