反射在JavaSwing编程中的应用 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/466/2021_2022__E5_8F_8D_E5_B0_84_E5_9C_A8J_c67_466992.htm 学习过Java Swing的读者一定对于Swing中相对较为复杂的事件驱动模型比较困惑，虽然事件驱动模型在Java Swing中被完完全全的体现出来了，但是对于一个软件初学者而言这样的近乎“ 裸体 ”的事件驱动模型确实是很难理解的。 Microsoft公司.Net框架与Java Swing的GUI编程相比要简单很多，同样是事件驱动模型.Net框架就进行了大量的封装处理，.Net把这种封装称之为委托器(Delegate)其代码如下： //当btnSubmit按钮被点击以后要求交给btnSubmit_Click方法处理// EventHandler在中间启到委托器的作用，//它负责将事件分发到指定的方法中进行处理this.btnSubmit.Click = new EventHandler(this.btnSubmit_Click).//事件处理方法// object sender:事件源，这里指btnSubmit对象// EventArgs e:事件处理参数，它保存了需要提供给程序员的必要信息private void btnSubmit_Click(object sender, EventArgs e){ //打印This is a button语句 System.Diagnostics.Debug.WriteLine("This is button").} 作为对比，我们来看看Java Swing的事件处理和委托就要复杂很多：代码如下：(您若还不是很了解Swing事件驱动的话，可以参考我的另外一篇文章：事件驱动模型实例详解（Java篇）)：//为btnSubmit增加侦听器SelectHandler,当btnSubmit被点击以后//有侦听器的actionPerformed负责处理该点击事件的业务//由于事件源btnSubmit和侦听器类SelectHandler处于两个不同的类中//为了让SelectHandler类取

得页面的信息，我们需要将窗体对象(this)//传入到侦听器中btnSubmit.addActionListener(new SelectHandler(this)).//侦听器SelectHandler，它必须实现动作事件ActionListener接口//以达到事件分发的作用class SelectHandler implements ActionListener { private CommonDialogDemo form = null. //将窗体对象CommonDialogDemo通过构造函数传入SelectHandler类中 public SelectHandler(CommonDialogDemo form) { this.form = form. } //事件处理方法，当btnSubmit被点击，自动执行以下打印代码 publicvoid actionPerformed(ActionEvent e) { System.out.println("This is button"). }} 根据以上代码，我们可以清晰的看到Java Swing要比.Net的麻烦的多，而且更不能让人忍受的就是，一个页面如果有多个按钮的话，我们必须针对每个按钮编写多个事件侦听类，而且这些类一般都会被设为内部类。学过软件建模的读者可能知道，内部在软件建模在软件工程中是不推荐使用的，所以这样的代码编写明显会增加设计冗余度和复杂度，因此我们可以考虑自己编写一个类似于.Net中EventHandler一样的事件委托类来处理事件分发。由于我们无权修改Java的编译器，所以我在这里将会借助于反射技术，利用一个事件委托类处理所有的点击事件，代码如下：package cn.softworks.teachersearchsystem.support.import java.awt.event.ActionEvent.import java.awt.event.ActionListener.import java.lang.reflect.Method./*** 该类是用来处理所有的Swing按钮点击事件，并根据将处理权*转交给使用者来处理**@authorChen.yu**/publicclass EventHandlerimplements ActionListener { //组件所在的窗体对象 private Object form = null. //受到委托的方法名 private String

methodName = null. /** *构造函数 * *@paramform 组件所在的窗体对象 *@parammethodName 受到委托的方法名 */ public EventHandler(Object form,String methodName) { this.form = form. this.methodName = methodName. } /** *事件处理委托方法 */ publicvoid actionPerformed(ActionEvent e) { //得到窗体对象的类型 Class formType = this.form.getClass(). try { //得到指定委托方法的类型 Method method = formType.getMethod(this.methodName, new Class[] {e.getClass()}). //调用指定的方法 method.invoke(this.form, new Object[] {e}). }catch(Exception ex) { return. } }} 现在我们来编写一个测试程序，代码如下：btnSearch.addActionListener(new EventHandler(this,"btnSearch_Click")).public void btnSearch_Click(ActionEvent e) { System.out.println("This is btnSearch").} 从以上代码中我们可以清晰的看到，事件处理和事件委托处于同一窗体中了，.Net方便的Delegate处理被我们用反射实现了。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com