Java桌面应用程序设计新贵：SWT简介 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/252/2021_2022_Java_E6_A1_8C_E9_9D_A2_c104_252404.htm Java语言的声望和它在桌面应用程序（GUI程序）所取得的成就显然极不相符，至今仍然很少能看到非常成功Java桌面程序。虽然有JBuilder，Netbean，JProbe等大型软件作为代表，但这仍不能证明Java的GUI程序是成功的：它们的外观总是和同一操作系统平台下的其它软件显得格格不入。对机器配置的需求也似乎永无止境，这使得它们只能被一些总是拥有当前最高性能PC的程序员们所容忍，或是那些不在乎金钱和时间的专业用户所接受。对绝大多数计算机使用者来说，AWT或SWING代表着怪异的界面和无法接受的速度。Standard Widget Toolkit（SWT）或许是Java这一噩梦的终结者，广大Java程序员终于可以开发出高效率的GUI程序，它们拥有标准的外观，几乎没有人能看出你的程序是用Java写出来的，更为重要的是，这些程序是跨平台的。 SWT本身仅仅是Eclipse组织为了开发Eclipse IDE环境所编写的一组底层图形界面 API。或许是无心插柳，或是有意为之，至今为止，SWT无论是在性能和外观上，都超越了SUN公司提供的AWT和SWING。目前Eclipse IDE已经开发到了2.1版本，SWT已经十分稳定。这里指的稳定应该包含两层意思： 一是指性能上的稳定，其中的关键是源于SWT的设计理念。SWT最大化了操作系统的图形构件API，就是说只要操作系统提供了相应图形的构件，那么SWT只是简单应用JNI技术调用它们，只有那些操作系统中不提供的构件，SWT才自己去做一个模拟的实现。可以看出SWT的性能上的稳定大

多时候取决于相应操作系统图形构件的稳定性。另一个稳定是指SWT API包中的类、方法的名称和结构已经少有改变，程序员不用担心由于Eclipse组织开发进度很快（Eclipse IDE每天都会有一个Nightly版本的发布），而导致自己的程序代码变化过大。从一个版本的SWT更新至另一版本，通常只需要简单将SWT包换掉就可以了。 第一个SWT程序 下面让我们开始一个SWT程序。（注意：以下的例子和说明主要针对Windows平台，其它的操作系统应该大同小异）。首先要在Eclipse安装文件中找到SWT包，Eclipse组织并不提供单独的SWT包下载，必须下载完整的Eclipse开发环境才能得到SWT包。SWT是作为Eclipse开发环境的一个插件形式存在，可以在${你的eclipse安装路径}\plugins路径下的众多子目录下去搜索SWT.JAR文件，在找到的JAR文件中包含了SWT全部的Java类文件。因为SWT应用了JNI技术，因此同时也要找到相对应的JNI本地化库文件，由于版本和操作平台的不同，本地化库文件的名称会有些差别，比如SWT-WIN32-2116.DLL是Window平台下Eclipse Build 2116的动态库，而在Unix平台相应版本的库文件的扩展名应该是.so，等等。注意的是，Eclipse是一个开放源代码的项目，因此你也可以在这些目录中找到SWT的源代码，相信这会对开发很有帮助。下面是一段打开空窗口的代码(只有main方法)。 import com.e2one.example.public class OpenShell{ public static void main(String [] args) { Display display = new Display(). Shell shell = new Shell(display). shell.open(). // 开始事件处理循环，直到用户关闭窗口 while (!shell.isDisposed()) { if (!display.readAndDispatch()) display.sleep(). } display.dispose(). }}

确信在CLASSPATH中包括了SWT.JAR文件，先用Javac编译例子程序。编译无错后可运行java -Djava.library.path=${你的SWT本地库文件所在路径} com.e2one.example.OpenShell，比如SWT-WIN32-2116.DLL件所在的路径是C:\swtlib，运行的命令应该是java -Djava.library.path=c:\swtlib com.e2one.example.OpenShell。成功运行后，系统会打开了一个空的窗口。 剖析SWT API 下面再让我们进一步分析SWT API的组成。所有的SWT类都用org.eclipse.swt做为包的前缀，下面为了简化说明，我们用*号代表前缀org.eclipse.swt，比如*.widgets包，代表的是org.eclipse.swt.widgets包。 我们最常用的图形构件基本都被包括在*.widgets包中，比如Button，Combo，Text，Label，Sash，Table等等。其中两个最重要的构件当数Shell和Composite。Shell相当于应用程序的主窗口框架，上面的例子代码中就是应用Shell构件打开一个空窗口。Composite相当于SWING中的Panel对象，充当着构件容器的角色，当我们想在一个窗口中加入一些构件时，最好到使用Composite作为其它构件的容器，然后再去*.layout包找出一种合适的布局方式。SWT对构件的布局也采用了SWING或AWT中Layout和Layout Data结合的方式，在*.layout包中可以找到四种Layout和与它们相对应的布局结构对象（Layout Data）。在*.custom包中，包含了对一些基本图形构件的扩展，比如其中的CLabel，就是对标准Label构件的扩展，上面可以同时加入文字和图片，也可以加边框。StyledText是Text构件的扩展，它提供了丰富的文本功能，比如对某段文字的背景色、前景色或字体的设置。在*.custom包中也可找到一个新的StackLayout布局方式。 SWT对用户操作的响应，比如鼠标

或键盘事件，也是采用了AWT和SWING中的Observer模式，在*.event包中可以找到事件监听的Listener接口和相应的事件对象，例如常用的鼠标事件监听接口MouseListener，MouseMoveListener和MouseTrackListener，及对应的事件对象MouseEvent。 *.graphics包中可以找到针对图片、光标、字体或绘图的API。比如可通过Image类调用系统中不同类型的图片文件。通过GC类实现对图片、构件或显示器的绘图功能。 对不同平台，Eclipse还开发了一些富有针对性的API。例如，在Windows平台，可以通过*.ole.win32包很容易的调用ole控件，这使Java程序内嵌IE浏览器或Word、Excel等程序成为可能！ 更复杂的程序 下面让我们展示一个比上面例子更加复杂一些的程序。这个程序拥有一个文本框和一个按键，当用户点击按键的时候，文本框显示一句欢迎信息。 为了文本框和按键有比较合理的大小和布局，这里采用了GradLayout布局方式。这种布局是SWT中最常用也是最强大的布局方式，几乎所有的格式都可能通过GradLayout去达到。下面的程序也涉及到了如何应用系统资源(Color)，以及如何释放系统资源。 private void initShell(Shell shell) { //为Shell设置布局对象 GridLayout gShellLay = new GridLayout(). shell.setLayout(gShellLay). //构造一个Composite构件作为文本框和按键的容器 Composite panel = new Composite(shell,SWT.NONE). //为Panel指定一个布局结构对象。 这里让Panel尽可能的占满Shell， 也就是全部应用程序窗口的空间。 GridData gPanelData = new GridData(GridData.GRAB_HORIZONTAL| GridData.GRAB_VERTICAL|GridData.FILL_BOTH).

panel.setLayoutData(gPanelData). //为Panel也设置一个布局对象。文本框和按键将按这个布局对象来显示。 GridLayout gPanelLay = new GridLayout(). panel.setLayout(gPanelLay). //为Panel生成一个背景色 final Color bkColor = new Color(Display.getCurrent(),200,0,200). panel.setBackground(bkColor). //生成文本框 final Text text = new Text(panel,SWT.MULTI|SWT.WRAP). //为文本框指定一个布局结构对象， 这里让文本框尽可能的占满Panel的空间。 GridData gTextData = new GridData (GridData.GRAB_HORIZONTAL| GridData.GRAB_VERTICAL|GridData.FILL_BOTH). text.setLayoutData(gTextData). //生成按键 Button butt = new Button(panel,SWT.PUSH). butt.setText("Push"). //为按键指定鼠标事件 butt.addMouseListener(new MouseAdapter(){ public void mouseDown(MouseEvent e){ //当用户点击按键的时候，显示信息 text.setText("Hello SWT"). } }). //当主窗口关闭时，会触发DisposeListener。这里用来释放Panel的背景色。 shell.addDisposeListener(new DisposeListener(){ public void widgetDisposed(DisposeEvent e) { bkColor.dispose(). } }).} 把这段代码中的方法initShell()加入到第一个打开空窗口的例子中，得到的是一段能成功运行的完整GUI应用程序。运行方法可参考第一个例子。