目录

[Riena与Nebula的比较 1](#_Toc417483741)

[从组件的实现与使用方式比较 1](#_Toc417483742)

[1 简单控件 1](#_Toc417483743)

[2 Dialog组件 2](#_Toc417483744)

[3 MessageBox消息提示 2](#_Toc417483745)

[4 Validation可校验组件（Text） 3](#_Toc417483746)

[从在组件加载方式角度上对比Riena与Nebula： 7](#_Toc417483747)

[总结： 11](#_Toc417483748)

# Riena与Nebula的比较

Riena与Nebula作为Elipse开源项目，都提供了丰富的SWT组件。Nebula项目侧重于为开发者提供方便易用的一系列组件，每个组件都位于独立的工程中，控件重用方便。Riena侧重于提供实现了Navigation树模型的框架。各个组件的使用都依赖于框架，控件不易重用。在两个开源项目中，对于常见的普通组件都直接使用SWT中定义的组件，例如：Button、Browser、ButtonCheck、ButtonRadio、ButtonToggle、DateTime、Date、CCombo、Combo、Table等等。从效果和使用上对比Nebula和Riena，对于简单组件Riena效果较好，但是在一些复杂组件却不如Nebula。

## 从组件的实现与使用方式比较

区别：Nebula控件库的重点提供了一系列复杂的组件，而Riena针对常用的组件和和窗体编写了 Renderer，进行重新渲染，外观上优于Nebula。

Riena通过实现IRidged接口，将所有组件封装成\*\*\*Ridget对象。

### 简单控件

Riena自定义了一系列简单组件，比如ImageButton。

ImageButton



ImageButton通过继承Composite实现了具有Hover等功能的ImageButton，通过setHoverImage(Image image)和setImage(Image image)等六个方法方法设置ImageButton在各种状态下的表现形式，并根据是否被点击改变状态。以下五张图片分别展示了ImageButton在点击前、点击前Hover、点击后、点击后Hover和被点击时的五种状态，并且当ImageButton处于disable状态时显示diaableImage。



在Nebula中没有类似的简单组件的定义。

### Dialog组件

在Riena中通过继承org.eclipse.jface.dialogs.Dialog，并且使用自定义的Render类org.eclipse.riena.ui.swt.RienaWindowRenderer对Dialog的样式重新绘制。



org.eclipse.riena.ui.ridgets.swt.views.AbstractDialogView以模板方法的形式重写buildView(final Composite parent)和createOkCancelButtons(final Composite parent)方法定义Dialog上内容区域和按钮区域的显示内容以及为按钮添加事件监听等操作。



### MessageBox消息提示

Riena通过MessageBox封装了一个集成自org.eclipse.jface.dialogs.MessageDialog的RienaMessageDialog类，重新绘制了MessageDialog的样式，功能与MessageDialog基本相同。并额为提供了自定义的按钮。





Nebula中没有定义自己的MessageDialog类。

### Validation可校验组件（Text）

Text被封装成了TextRidget类，TextRidget还实现了Imarkable接口表示TextRidget能够被标记，通过调用addValidationRule(IValidator validator, ValidationTime validationTime)方法传入校验的正则表达式和校验时机，如果不符合校验条件，则调用addMarker(IMarker)方法为Text添加红色边框或者改变文本内容颜色。





1. MasterDetailsComposite。MasterDetailsComposite是Riena中提供的用于查看表格信息以及表格中每一行的行对象的详细信息的复杂组件。还提供了对表中的行对象进行编辑的功能。





1. 进度条

Riena通过继承org.eclipse.swt.widgets.Canvas定义了进度条StatusMeterWidget图中所示上面的为org.eclipse.swt.widgets.ProgressBar，下面的为Riena定义的StatusMeterWidget。





1. InfoFlyout推送消息窗口。

Riena自定义了InfoFlyout抽象类，InfoFlyoutRCP继承自InfoFlyout抽象类，通过调用InfoFlyout对象的openFlyout()在桌面上产生一个消息推送窗口，并且调用setPositionCorrectionY(**final** **int** positionCorrectionY)方法设置推行窗口的位置。





1. 在Nebula中提供了一系列复杂的组件。Nebula中的复杂组件大多是继承一个Canvas，并在Canvas上一GC的方式绘制组件内容。
* Gallery

在Nebula中提供了用于分组显示缩略图的组件，可用来实现照片预览和文件导航器。



* GanttChart甘特图

Nebula中实现了比较复杂的甘特图



* Oscilloscope示波器

Nebula中的Oscilloscope组件以直观的方式监视实时的动态变化。



* PGroup可折叠的分组组件，为用户将一些在逻辑上属于同一类的组件放在一个Group下，向用户提供有意义的信息。



* Pagination表格分页

当表格行数过多时，Pagination自动分页，并提供向前向后导航功能。



* RadioGroup

实现了一个组织Radio的组件，而不用每次为Radio设置Group信息。



* TableCombo一个展开内容为表格的下拉菜单



* XViewer

在TreeView的基础上，提供更加动态的具有筛选排序等功能的电子表格，并提供自定义表格



* GeoMapViewwe

Nebula提供了可拖动的地图组件

## 从在组件加载方式角度上对比Riena与Nebula：

Nebula中采用了SWT普通的组件生成与加载方式，在组件生成时传入parent，并且每个组件包可以独立使用。Riena中将单个组件使用MVC模式将管理一系列添加到Composite上的组件，得到一个ViewPart。各个ViewPart以OSGI方式加载到Example Shell的相应位置。

Riena使用了一个Navigation模型树的概念，根节点是创建应用的Application类，在根节点下面包含了至上而下sub-application、module group、module、sub module四层。并且各层次关系如下

* sub-application
	+ module group
		- module
			* sub-module
			* sub-module
		- module
			* sub-module
	+ module group
		- module
			* sub-module
* sub-application
	+ module group
		- module
			* sub-module



在Rinea的Demo中各层分别对应如下：

1.sub-application



2.mudule group、mudules和sub module



Riena提供了三种方式向Navigation树插入一个Navigation节点：

* + - * 1. Programmatic Creation
				2. Extension Points
				3. Navigation Node Assemblers

其中每个ViewPart对应一个Navigation树的叶子节点的SubModule，并使用一个唯一的ID相互绑定。当一个叶子节点的SubModule被选中，则加载绑定的ViewPart。每个ViewPart遵循Model-View-Controller（MVC）模式。其中View只负责创建和展示一系列SWT的组件，不定义SWT组件的行为和数据；Model为SWT组件提供数据；在Controller实现SWT的所有行为，并负责联系V与M。

* View

Riena中的View 继承自org.eclipse.riena.navigation.ui.swt.views. SubModuleView，用来声明一个Composite上的组件，并且Composite上的所有Widget都用addUIControl(final Object uiControl, final String bindingId) 方法将传入一个id将View和Controller关联在一起。



@Override

 protected void basicCreatePartControl(final Composite parent) { parent.setBackground(LnfManager.getLnf().getColor(LnfKeyConstants.SUB\_MODULE\_BACKGROUND));

 parent.setLayout(new GridLayout(2, true));

 UIControlsFactory.createLabel(parent, "Text Field:"); //$NON-NLS-1$

 final Text textField = UIControlsFactory.createText(parent);

 addUIControl(textField, "textField"); //$NON-NLS-1$

}

* Model

Model是用来记录对应UI的相关的Data

* Controller

Controller继承自org.eclipse.riena.navigation.ui.controllers.NavigationNode

Controller<ISubModuleNode>，用来定义ViewPart上各个组件的行为和逻辑，通过getRidget(final Class<R> ridgetClazz, final String id)方法得到组件，并对其行为逻辑定义。



@Override

 **public** **void** configureRidgets() {

 **super**.configureRidgets();

 messageBox = getRidget(IMessageBoxRidget.**class**, "messageBox"); //$NON-NLS-1$

 messageBox.setTitle(getNavigationNode().getLabel());

 messageBox.setText("Change value in the previous sub-module and \ntry it again."); //$NON-NLS-1$

 messageBox.setOptions(IMessageBoxRidget.*OPTIONS\_OK*);

 messageBox.setType(IMessageBoxRidget.Type.*INFORMATION*);

 }

# 总结：

经过对于Nebula和Riena的比较，Nebula在对于较复杂的组件实现上较为有优势，而Riena通过Render重新渲染提供的简单组件更加美观。