目录

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

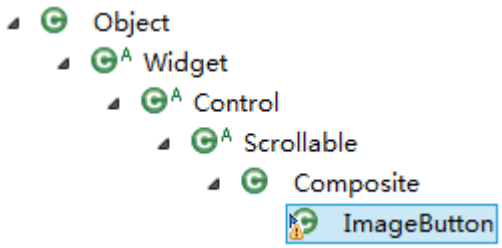
区别：Nebula 控件库的重点提供了一系列复杂的组件，而 Riena 针对常用的组件和和窗体编写了 Renderer，进行重新渲染，外观上优于 Nebula。

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

1 简单控件

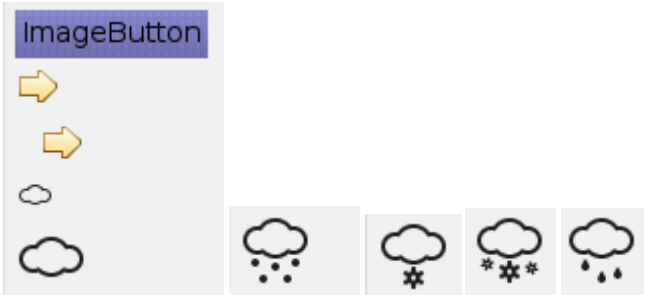
Riena 自定义了一系列简单组件，比如 ImageButton。

ImageButton



ImageButton 通过继承 Composite 实现了具有 Hover 等功能的 ImageButton，通过

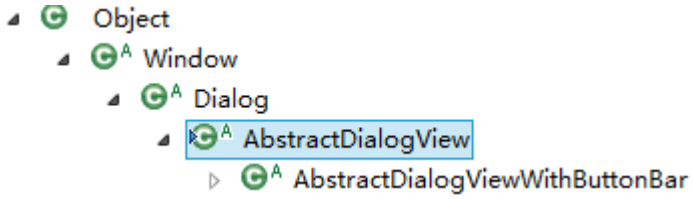
setHoverImage(Image image) 和 setImage(Image image) 等六个方法方法设置 ImageButton 在各种状态下的表现形式，并根据是否被点击改变状态。以下五张图片分别展示了 ImageButton 在点击前、点击前 Hover、点击后、点击后 Hover 和被点击时的五种状态，并且当 ImageButton 处于 disable 状态时显示 diaableImage。



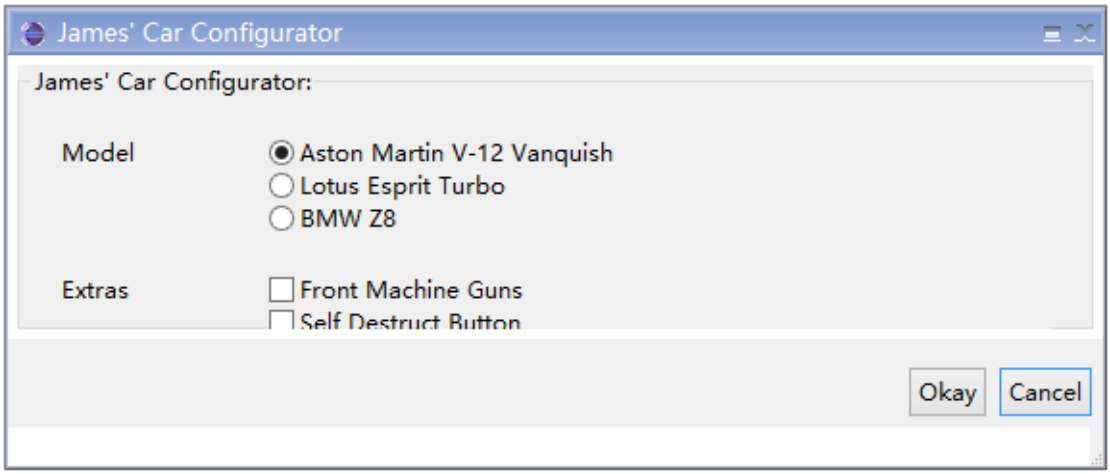
在 Nebula 中没有类似的简单组件的定义。

2 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。

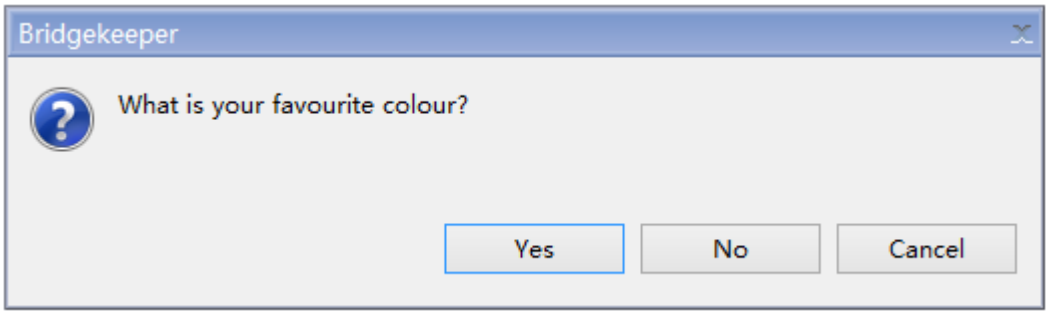


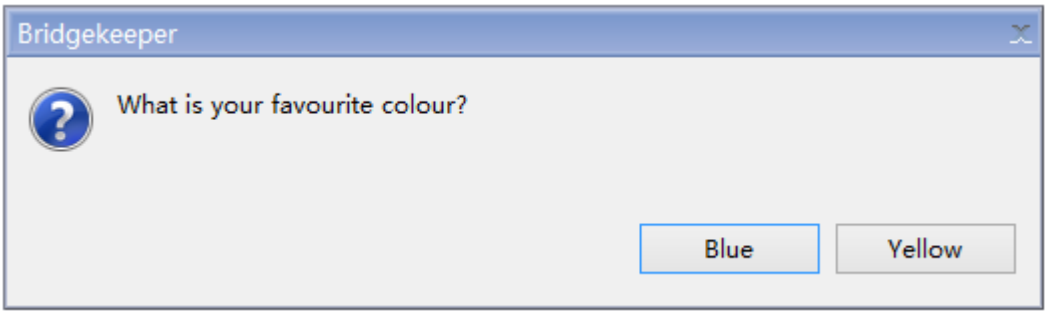
`org.eclipse.riena.ui.ridgets.swt.views.AbstractDialogView` 以模板方法的形式重写 `buildView(final Composite parent)` 和 `createOkCancelButtons(final Composite parent)` 方法定义 `Dialog` 上内容区域和按钮区域的显示内容以及为按钮添加事件监听等操作。



3 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。

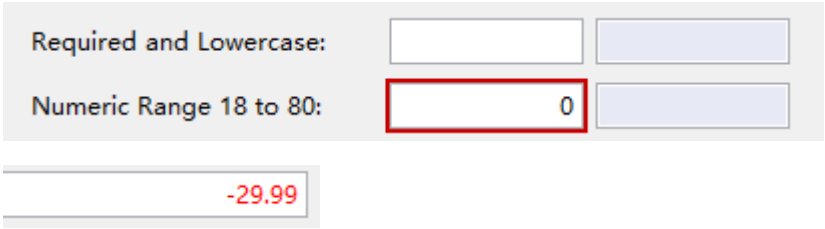




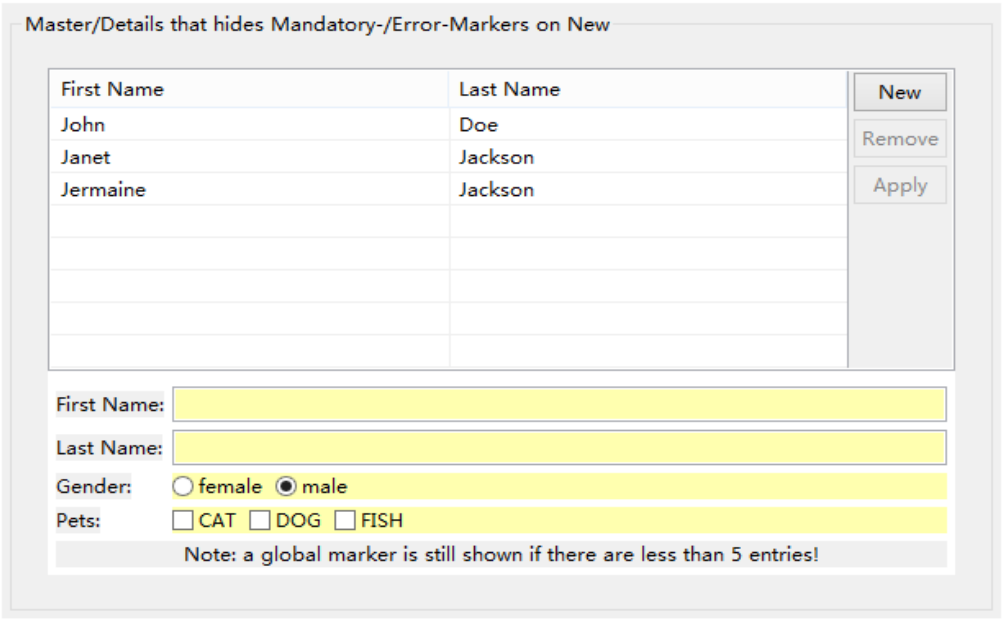
Nebula 中没有定义自己的 MessageDialog 类。

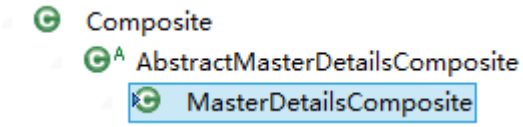
4 Validation 可校验组件（Text）

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。



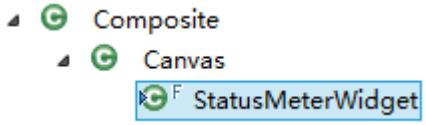
1. MasterDetailsComposite。MasterDetailsComposite 是 Riena 中提供的用于查看表格信息以及表格中每一行的行对象的详细信息的复杂组件。还提供了对表中的行对象进行编辑的功能。

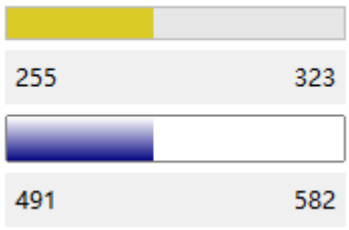




2. 进度条

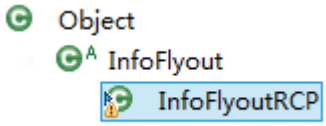
Riena 通过继承 org.eclipse.swt.widgets.Canvas 定义了进度条 `StatusMeterWidget` 图中所示上面的为 org.eclipse.swt.widgets.ProgressBar，下面的为 Riena 定义的 `StatusMeterWidget`。

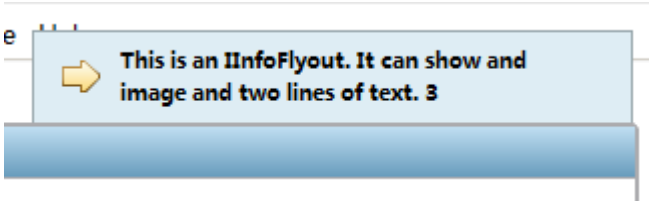




3. `InfoFlyout` 推送消息窗口。

   `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。





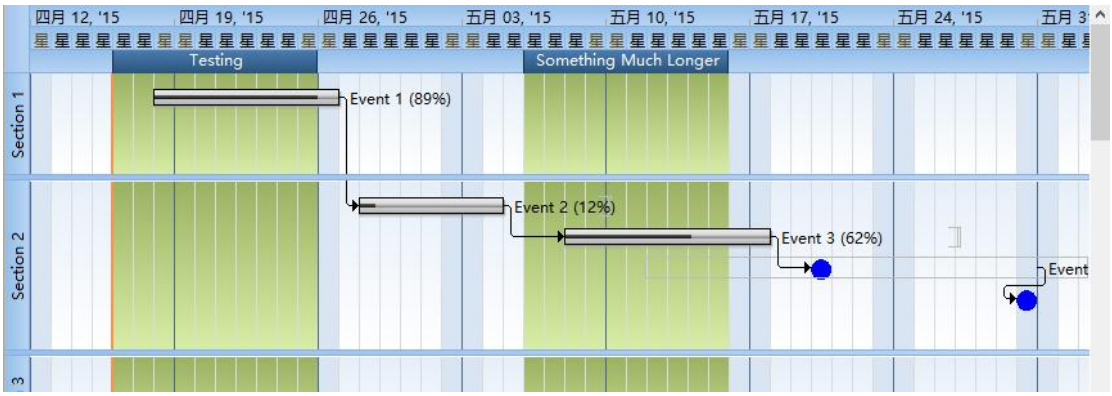
4. 在 Nebula 中提供了一系列复杂的组件。Nebula 中的复杂组件大多是继承一个 Canvas，并在 Canvas 上一 GC 的方式绘制组件内容。

   - 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。

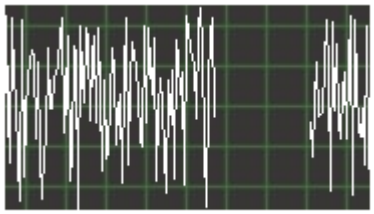
   - `GanttChart` 甘特图

   Nebula 中实现了比较复杂的甘特图



- `Oscilloscope` 示波器

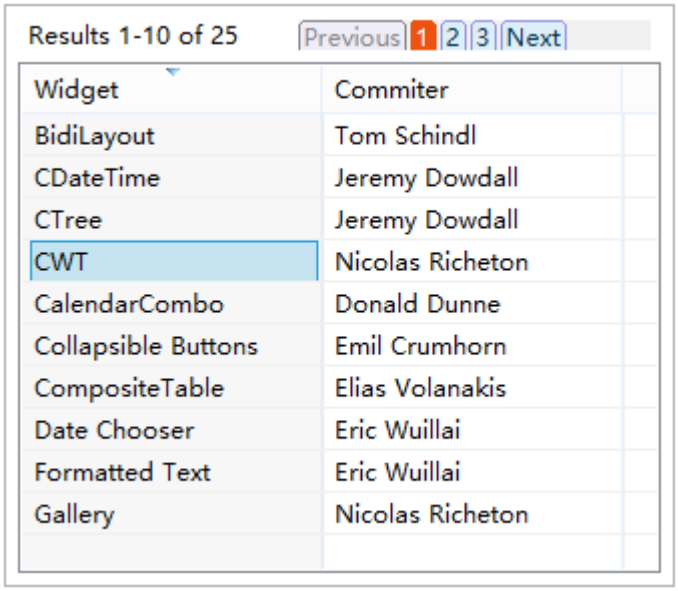
  Nebula 中的 `Oscilloscope` 组件以直观的方式监视实时的动态变化。



- 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。

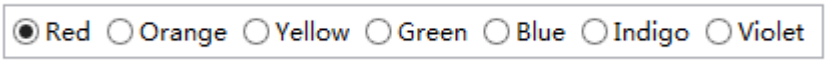
- Pagination 表格分页

  当表格行数过多时，Pagination 自动分页，并提供向前向后导航功能。

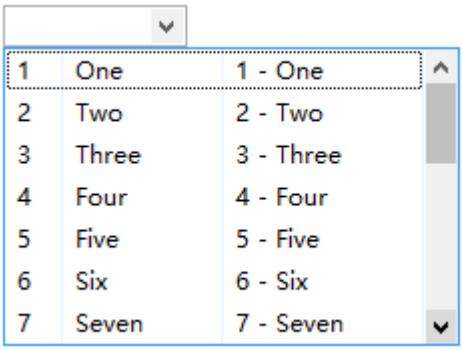


- RadioGroup

  实现了一个组织 Radio 的组件，而不用每次为 Radio 设置 Group 信息。

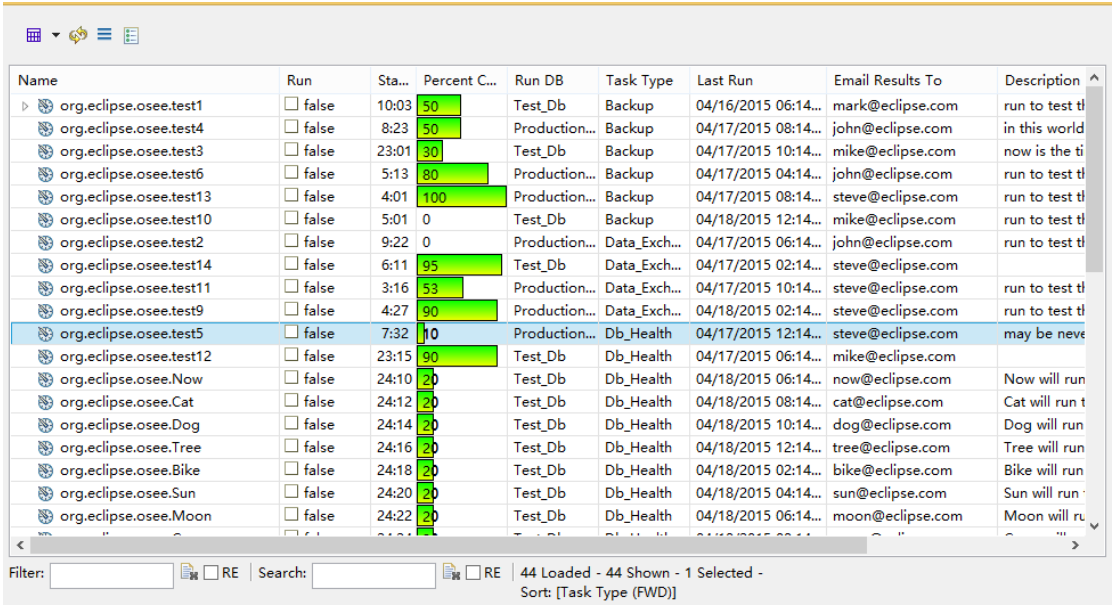


- TableCombo 一个展开内容为表格的下拉菜单



- 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

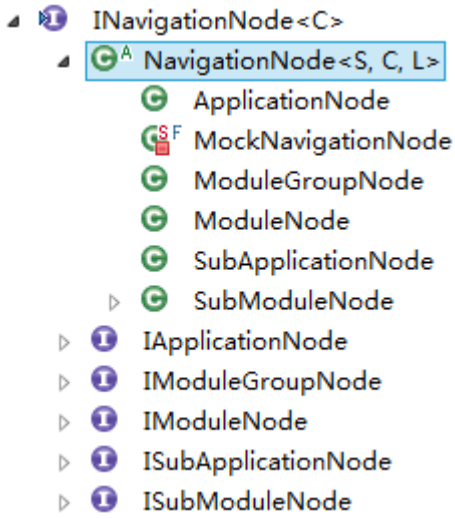
从在组件加载方式角度上对比 Riena 与 Nebula：

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

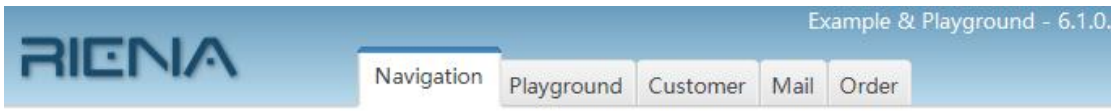
- sub-application
  - module group
    - module
      - sub-module
      - sub-module
    - module
      - sub-module
  - module group
    - module
      - sub-module

- sub-application
    - module group
        - module
            - sub-module

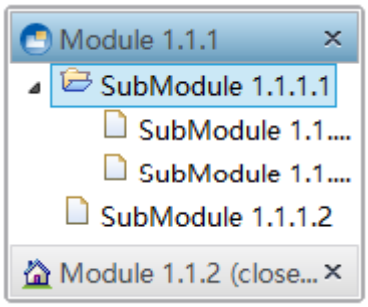


在 Rinea 的 Demo 中各层分别对应如下：

1. sub-application



2. mudule group、mudules 和 sub module



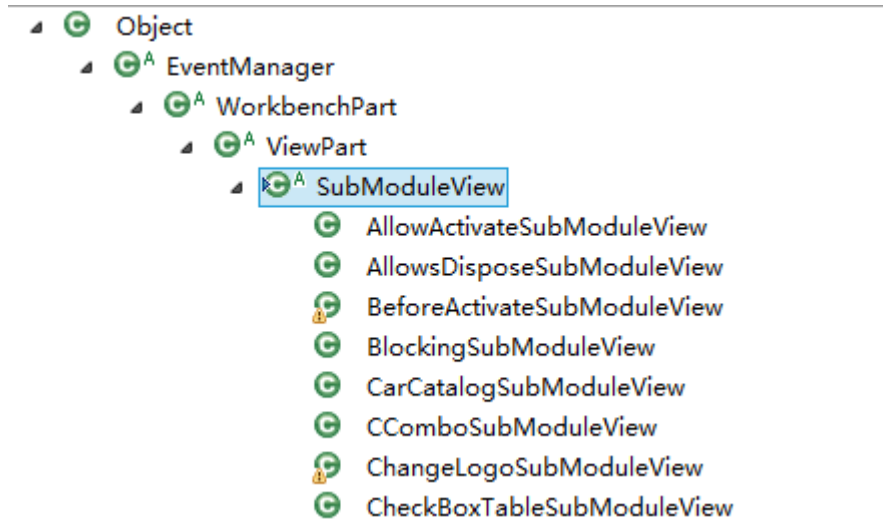
Riena 提供了三种方式向 Navigation 树插入一个 Navigation 节点：

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"); //$NON-NLS-1$
}
```

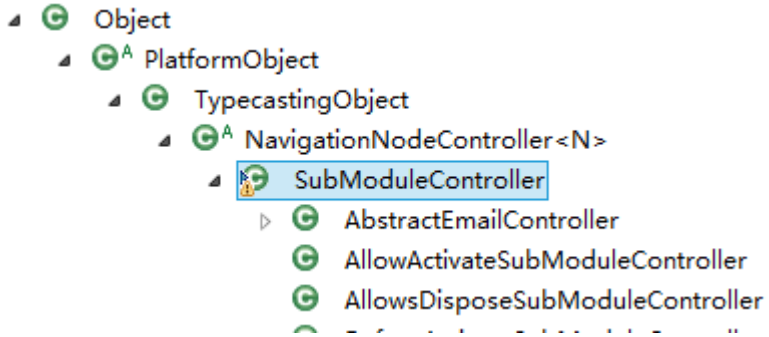
- Model

  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。