EXTENSIBLE PROPERTIES EDITOR FRAMEWORK SPECIFICATION

Author: Qiangsheng Wang, Yueqian Wang, Chao Chen

1. Introduction

The properties editing view provide a widget based page to let user change the properties' value of selection. By collection the new requirements from community, we redesign the whole structure of this view from technical side to look and feel side.

The requirement includes:

- **Different page content based on selection context.** The page for the same type of selection can have different content, which bases on the context of the selection. For example, the element from library will have library information field but local element doesn't have.
- **Different action behavior based on selection context.** The user actions behavior is different depends on selection context. For example, some items don't support Total function in Hyperlink builder but some others supports. Even they share the same Hyperlink page but their need rewrite their own click action to pop up the expression builder with different function tree.
- **Different visual effect based on property.** When the page shows the value of selection, the same property can have different visual effect. For example, the properties show as grey or white that depends on if the multiple selections contains the same type of elements.
- **Decouple the page with model properties definition.** By original design of this view, we assume all properties are from an element handle. Now, we all know this assumption is not true. Some are not properties (Here I mean the properties defined in BIRT ROM) with the other are properties of structure element.
- **Extensible and customizable by programmers.** This is a general requirement and the real requirements are from the extended item. The Chart item has similar but different property pages. They wish we can provide way to let them reuse the code but customized some logic by their request.

2. APIs

2.1 Classes Definition

2.1.1 public class AttributeView extends PageBookView

This is the main class for the attribute view. This is BIRT standard view has id "org.eclipse.ui.views.PropertySheet".
Attribute view page is discovered by the attribute view automatically when a part is first activated. The attribute view asks the active part for its attribute page; this is done by invoking **getAdapter(AttributeView.class)** on the part.

2.1.2 public abstract class AttributeViewPage extends Page

The standard implementation of attribute view page which presents the properties and values obtained from the current selection. This page organized with Tab style by using **CTabFolder**.

2.1.3 public interface IPageGenerator

This is interface provides the service to manage the tab pages in the CTabFolder of attribute view page.

void createTabItems(CTabFolder tabFolder, Object input);

The only method in the interface that serves to creates tab pages for the selection items and added them into **CTabFolder**.

2.1.4 public class DefaultPageGenerator implements IPageGenerator

This is the standard implementation of **IPageGenerator** which provides the lifecycle management of tab pages.

2.1.5 public interface IPropertyTabUI

This is the class provides the basic interface of tab pages. Tab pages are the pages mange by **CTabFolder**.

public void buildUI(Composite parent);

This method creates and layouts the widget on the page.

public String getTabDisplayName( );

This method returns the name of this page. This name shows on the tab area of page.

public void setInput(Object elements);

This method sets the input to this page. Usually the input is a list that contains a group of selection elements.

public void dispose( );

This method releases all sources of this page. This method called when the properties edit view disposed.

2.1.6 public abstract class Section

This is the attribute section class that manages a group of **IPropertyDescriptor** which provides a single function, for example the group of **border setting buttons** to provide border setting function. This class provides the life cycle management of these widgets.

Every Section class must have a unique ID so that user can overwrite the section by provide associate the new implementation with defined ID.

2.1.7 public interface IPropertyDescriptor

This is the interface to serve as the manager of widget and associated provide. It controls the full life cycle of widget.

Control createControl(Composite parent);

Create widget and added it on the parent.

void load( );

Load values and set it to the widget.

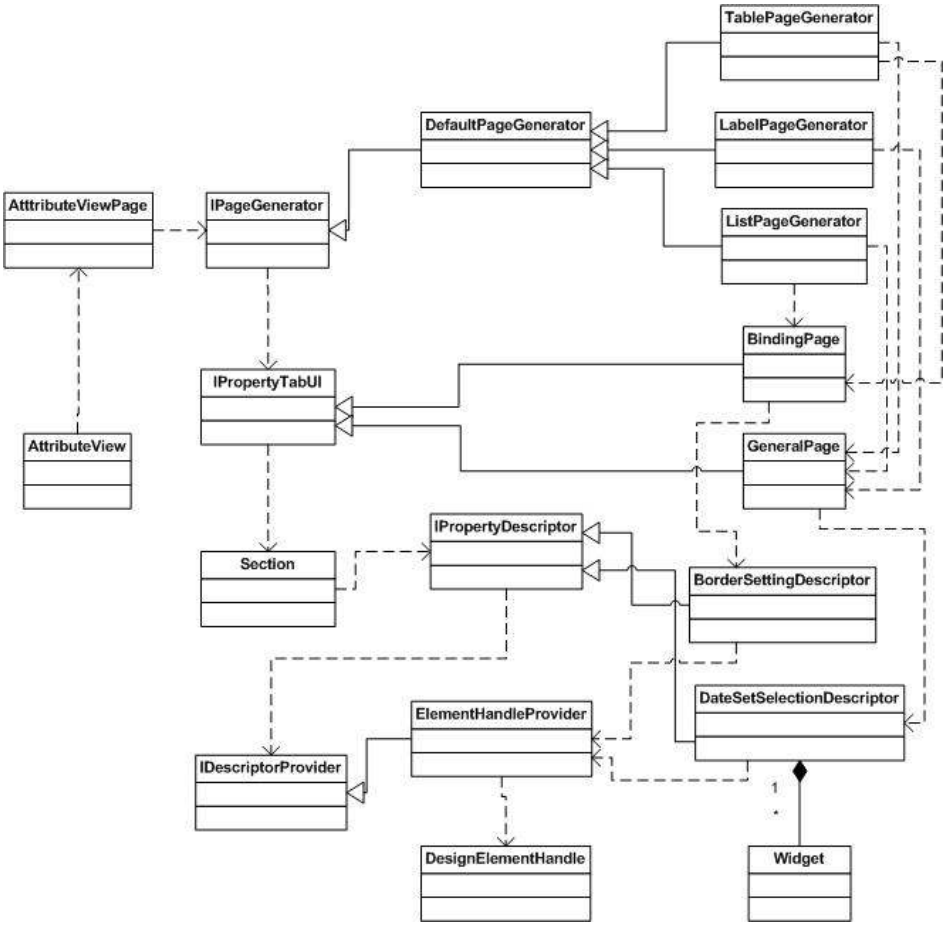
void save();

Save the value on the widget to the associated model by **IDescriptorProvider** interface**.**

2.1.8 public interface IDescriptorProvider

That's the interface to provide logic to access model of widget. The **IDescriptorProvider** is called by **IPropertyDescriptor**.

2.1.9 Class diagram



2.2 Extension point definition

2.2.1 Change the extension point definition structure.

Please see following code, the class, icons and other definition are all parallel to the model extension.

```
<extension
   id="chart"
   name="The chart extension"
```

```
  point="org.eclipse.birt.report.designer.ui.reportitemUI">
<model extensionName="Chart"/>
                <reportItemFigureUI
                        class="org.eclipse.birt.chart.reportitem.ui.ChartReportItemUIImpl"/>
                <builderclass="org.eclipse.birt.chart.reportitem.ui.ChartReportItemBuilderImpl"/>
          <outline icon="icons/obj16/cell.gif"/>
                <palette icon="icons/pal/palette_icon.gif"
       category="Content"/>
          <editor
        showInMasterPage="false"
       canResize="false"
       showInDesigner="true"/>
          <outline
                  icon="icons/pal/palette_icon.gif"/>
                <propertyPage
                        class="org.eclipse.birt.chart.reportitem.ui.ChartReportItemPropertyEditUIImpl"/>
        </extension>
```

Suggest changing as the following. This move those definition into the model extension. By these changes, on reportitemUI extension point can contains several model definitions.

```
<extension
  id="chart"
  name="The chart extension"
  point="org.eclipse.birt.report.designer.ui.reportitemUI">
<model
                 extensionName="Chart">
<reportItemFigureUI
                class="org.eclipse.birt.chart.reportitem.ui.ChartReportItemUIImpl"/>
<builder
      class="org.eclipse.birt.chart.reportitem.ui.ChartReportItemBuilderImpl"/>
<outline icon="icons/obj16/cell.gif"/>
<palette
       icon="icons/pal/palette_icon.gif"
       category="Content"/>
<editor
       showInMasterPage="false"
       canResize="false"
       showInDesigner="true"/>
<outline
                 icon="icons/pal/palette_icon.gif"/>
<propertyPage class="org.eclipse.birt.chart.reportitem.ui.ChartReportItemPropertyEditUIImpl"/>
        </extension>
</model>
```

3. Potential changes

File configuration

By previous design, the attribute view can be configuration by files. We can define the pages, sections and widgets by external file, for example plugin.xml. All the pages can be generated by a common framework. The question is if this needs to do.

From view of me, only reason is that we assume the pages changed frequently. By this configuration way, we don't need change code but can change the view's looks and feel quick. I leave this as TBD and wait feedbacks.