<!--QiQu-->

Property-Dialogs In Depth

aloba
Softwareentwicklung + Consulting

SOURCEFORGE™
.net

Table of Contents

A simple example

Let's start with an easy example. We'd like to create a property dialog containing a label and File-Chooser widget. So, open your xml-editor (e.g. the eclipse-plugin XMLBuddy) and configure it so that it will be using the **propertydialog.dtd** for examlpe files with the root-tag "propertydialog". Done this, let's write our first dialog configuration file:

```
<propertydialog title="Small Gui Example">
        <widgetlist name="modelselection">
                <widget type="label"
                        name="Select the model"
                        propertyname=""
                        startvalue=""/>
                <widget type="file"
                        name="Model"
                        propertyname="model"
                        startvalue="/model/defaultmodel.xml"
                        helptext="Select the model that should be loaded"/>
        </widgetlist>
</propertydialog>
```

The main element is the **propertydialog** element. It's attribute **title** defines the title that is displayed in the title bar of the dialog. Widgets have to be defined inside a **widgetlist** element. the attribute **name** defines the groupname of the widget under the widgetlist. It is needed in order to persist the entered information in a file. Widgets are defined with the **widget** element. The attribute **type** defines the type of the widget (label, file, dir, date, text, ...). A complete list with all possible widget types can be found at the end of this tutorial. The attribute **name** defines the text that is displayed for the widget. The content is stored under the property name defined in the attribute **propertyname**. It is possible to define an initial value with the attribute **startvalue**. The text that is defined in the attribute **helptext** is displayed as tooltip for the widget.

Now let's have look, how the dialog can be openend and how the entered values can be accessed. We actually just need two lines:

```
<LoadDoc FileName="'script/gui/SimpleDialog.xml'" NewDocRef="$simpleDialogDescription"/
<ShowPropertyDialog GuiNodeRef="$simpleDialogDescription" PropFileName="'data/dialogcon
```

Our dilaog definition is an XML-file. So simply load it with the **LoadDoc** command. The command **ShowPropertyDialog** opens the dialog. The attribute **GuiNodeRef** must be defined. It contains the XML-Node with the dialog definition. The second attribute (**PropFileName**) is optional. If it is defined, the content of the dialog are stored inside this file. This way, the content is not lost between multiple execution of the script. By the way, this file is XML-as well. It contains the same structure as the definition of the property dialog. You could also read that file inside the script itself.

When we define a widget, we define the attribute **propertyname**. The value of the widget is stored under this property name. Inside the script, you can simply access it by its' name.

```
<EchoText InfoText="'Entered model is: ' + model"/>
```

Property dialogs have an ok and a cancel button. To see, which of those button had been pressed, properties are created. If ok is pressed, the property **okbutton** is set to 'true' and the property **cancelbutton** is set to false. If the cancel button is pressed, the properties are set accordingly. With that knowledge, we write our final version of our tescript:

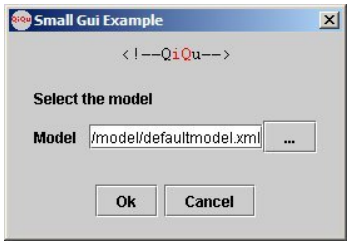
```
<QiQuScript>
        <LoadDoc FileName="'script/gui/SimpleDialog.xml'" NewDocRef="$simpleDialogDescr
        <ShowPropertyDialog GuiNodeRef="$simpleDialogDescription" PropFileName="'data/d

        <If Condition="cancelbutton">
```

```
                <EchoText InfoText="'Cancel button pressed!'"/>
        </If>

        <If Condition="okbutton">
                <EchoText InfoText="'Ok button pressed'"/>
                <EchoText InfoText="'Entered model is: ' + model"/>
        </If>
</QiQuScript>
```

If you run the script, the following dialog should appear:

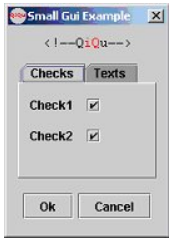


Tabbed Panes

Of course, QiQu-PropertyDialogs do also support tabbed panes. The following example will show you, what is possible.

Simple Tabbed Panes

```
<propertydialog title="Small Gui Example">
  <tablist name="mainTabs">
    <tab name="Checks">
      <widgetlist name="checkboxes">
        <widget type="check" name="Check1" propertyname="check1" startvalue="true"/>
        <widget type="check" name="Check2" propertyname="check2" startvalue="true"/>
      </widgetlist>
    </tab>
    <tab name="Texts">
      <widgetlist name="texts">
        <widget type="text" name="Text1" propertyname="text1" startvalue="Text 1"/>
        <widget type="text" name="Text2" propertyname="text2" startvalue="Text 2"/>
      </widgetlist>
    </tab>
  </tablist>
</propertydialog>
```

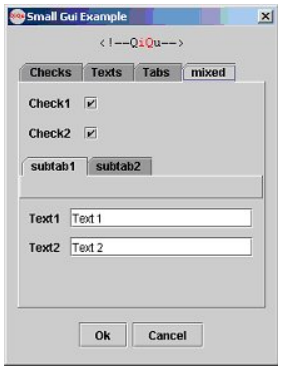


Nested Tabbed Panes

```
<propertydialog title="Small Gui Example">
  <tablist name="mainTabs">
    ...
    <tab name="Tabs">
      <tablist name="subtabs">
        <tab name="subtab1">
        </tab>
        <tab name="subtab2">
        </tab>
      </tablist>
    </tab>
  </tablist>
</propertydialog>
```

Nested Tabbed Panes with mixed content

```
<propertydialog title="Small Gui Example">
  <tablist name="mainTabs">
    ...
    <tab name="mixed">
      <widgetlist name="checkboxes">
        <widget type="check" name="Check1" propertyname="check1" startvalue="true"/>
        <widget type="check" name="Check2" propertyname="check2" startvalue="true"/>
      </widgetlist>
      <tablist name="subtabs">
        <tab name="subtab1">
        </tab>
        <tab name="subtab2">
        </tab>
      </tablist>
      <widgetlist name="texts">
        <widget type="text" name="Text1" propertyname="text1" startvalue="Text 1"/>
        <widget type="text" name="Text2" propertyname="text2" startvalue="Text 2"/>
      </widgetlist>
    </tab>
  </tablist>
</propertydialog>
```



Changing widgets and content at runtime

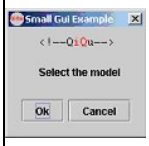
We've mentioned it already. With QiQu, most things are XML. The language, the input, the output, the property dialog definition, and so on, and since the main purpose of the QiQu language is the transformation of XML, we are able to change the mentioned parts. This is pretty handy, if we'd like to adjust the content of a property dialog at runtime.

Let's say, we want to create sourcecode based on a class model. A required feature is, that the user of the generator is able to define which classes should generated. Simply define a propertydialog with a list widget. Inside the script, load the dialog definition and add the appropriate entries at runtime.

As a matter of fact. Nobody forces you to define your property definition files. You could create the needed Xml structure at runtime with a QiQu script itself.

PropertyDialog Widgets

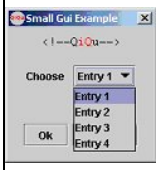
widgettype	screenshot	widget defintion

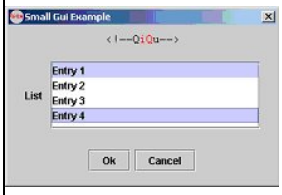
label		`<widget type="label"`
		`name="Select the model"`
		`propertyname=""`
		`startvalue=""`
		`helptext=""/>`

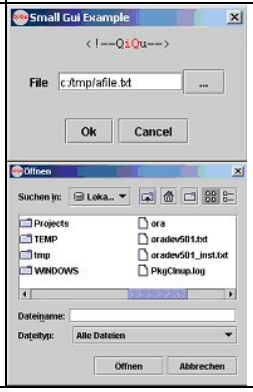
check		`<widget type="check"`
		`name="Please check"`
		`propertyname="checkedvalue"`
		`startvalue="true"`
		`helptext="Checkbox helptext"/>`

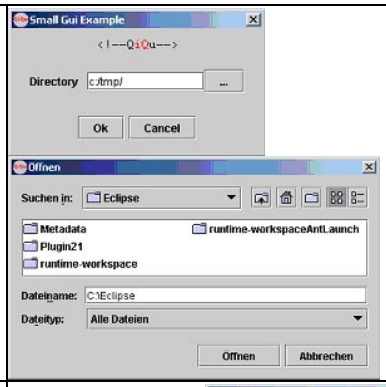
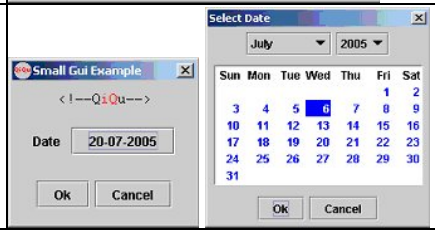
text		`<widget type="text"`
		`name="Text"`
		`propertyname="Text"`
		`startvalue="sometext"`
		`helptext="Text helptext"/>`

textarea		`<widget type="textarea"`
		`name="Textarea"`
		`propertyname="Textarea"`
		`startvalue="sometext more text"`
		`helptext="Textarea helptext"/>`

combo		`<widget type="combo"`
		`name="Choose"`
		`propertyname="combo"`
		`startvalue="Entry 1"`
		`helptext="Combo helptext">`
		`<item value="Entry 1"/>`
		`<item value="Entry 2"/>`
		`<item value="Entry 3"/>`
		`<item value="Entry 4"/>`
		`</widget>`

list		`<widget type="list"`
		`name="List"`
		`propertyname="list"`
		`startvalue="Entry 1"`
		`helptext="List helptext">`
		`<item value="Entry 1"/>`
		`<item value="Entry 2"/>`
		`<item value="Entry 3"/>`
		`<item value="Entry 4"/>`
		`</widget>`

file		`<widget type="file"`
		`name="File"`
		`propertyname="file"`
		`startvalue="c:/tmp/afile.txt"`
		`helptext="File selection helptext"/>`

dir		<widget type="dir" name="Directory" propertyname="directory" startvalue="c:/tmp/" helptext="Directory selection helptext"/>
date		<widget type="date" name="Date" propertyname="date" startvalue="20-07-2005" helptext="Date selection helptext"/>