Notebook Widget

The Notebook widget is the most complex widget, and hence often overlooked as a user interface widget. In this module we describe how to create a complex notebook widget, by providing an example and exercises.

1. Introduction

The Notebook widget is a composite widget containing a Subcanvas and up to two rows of tabs. The model for each row of tabs is an instance of SelectionInList — its list is often a collection of Symbols. The aspect for each row of tabs is specified in the Properties Tool (see Fig.1). One row is known as *Major* and the other as Minor. The inner area of a Notebook widget is a Subcanvas whose specification can be modified at run–time.

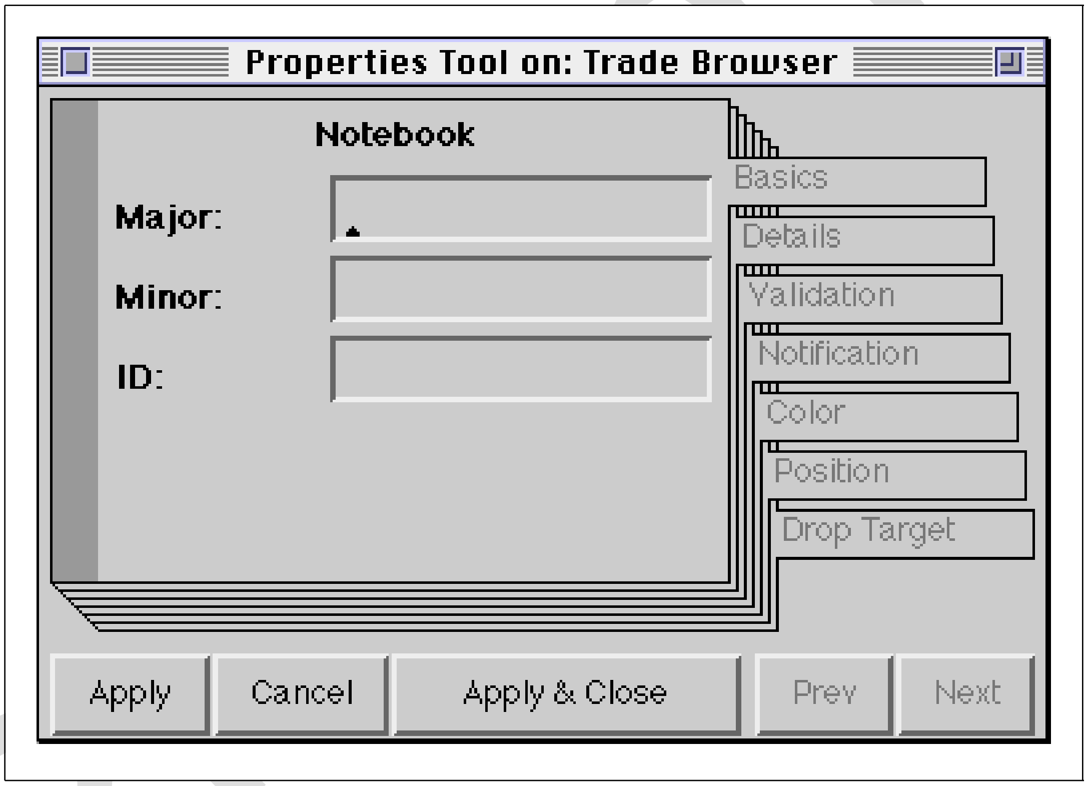

Figure 1: Notebook Properties

The most complex use of the Notebook widget is to change the Subcanvas based on the tab selection. However, we'll start with something simpler: using the major tab selection to control the contents of a Dataset widget.

2. Major Tabs

The Notebook, in common with all VisualWorks widgets, is specified using the Properties Tool. The minimum requirements is to specify the *Major* aspect: the message that is sent to the application model to return an instance of SelectionInList.

Ex 1. Open a new Canvas and add a Notebook widget. Give it a *Major* aspect of #majorKeys and an *ID* property of #notebookID. Install the Canvas in a new class named TradeBrowser, then use the Definer to create the model for the Notebook. Create an instance of your new class to ensure that it works.

The contents of the tabs are usually instances of Symbol or String. For this example, we will display the names of traders in the tabs.

Ex 2. Browse BondTrade class to discover how to retrieve a random collection of instances of BondTrade. Add an initialize method to class TradeBrowser, which causes the tabs to display the names of traders. Note that selecting a tab has no effect.

Initially, the inner area of the Notebook widget (the Subcanvas) is unassigned and no tab is selected. To display an initial Subcanvas, add a postOpenWith: method to assign the Subcanvas and possibly set the tab selection.

A Subcanvas is assigned by sending the following message to the Subcanvas widget:

client: *aModel* spec: *aSpec*

where *aSpec* is the Canvas specification to be used and *aModel* is an instance of the class in which it is defined.

To programmatically select one of the row of tabs, set the selection of its SelectionInList model using the selection: or the selectionIndex: message.

Ex 3. Open a new Canvas, and add a Dataset widget to it. Specify its properties so that it can display (read–only) the attributes of a BondTrade. Install the Canvas in class TradeBrowser (giving it the name #tradeSpec), and use the Definer to create its models. Add a postOpenWith: method to class TradeBrowser, causing the Dataset widget to be displayed when the application is first opened. The Dataset widget should display all of the instances of BondTrade that you accessed in Ex.2.Similarly, add a message expression to the method so the first tab is selected.

Usually you want to trigger some behavior when a tab is selected. This may be achieved by using the onChangeSend:to: message, sent to the selectionIndexHolder of the SelectionInList. The message is usually sent in the initialize method:

```
selectionIndexHolder onChangeSend: #changedTab to: self
```

where changedTab is the name of the message to be sent when the tab selection is changed. In our example, we wish to change the contents of the Dataset. However, this technique can also be used to replace the Subcanvas, add or remove minor key tabs.

Ex 4. Add the appropriate message expressions to the initialize method of class TradeBrowser (and any other methods you think necessary), so that when a tab is selected the Dataset contains the appropriate instances of BondTrade are displayed for the selected Trader.

3. Minor Tabs

Minor tabs may be associated with individual major tabs. For example, selecting a major tab might cause a new set of minor tabs to appear. Alternatively, minor tabs may represent another dimension of information for the Subcanvas.

In this example, we'll associate the minor tabs with the selected major tab: the minor tabs will display the locations of the trades associated with a selected major tab (i.e., a selected trader).

Ex 5. Modify the properties of the Notebook widget so that it specifies a *Minor* aspect: #minorKeys. Re–install the Canvas and use the Definer to modify the class definition and create an accessing method for the new aspect.

Ex 6. Modify the initialize method of class TradeBrowser (and add any other necessary methods) so that when a major tab is selected, the minor tabs display the possible locations for that trader. Similarly when a minor tab is selected, the Dataset widget should only contain instances of BondTrade appropriate to the selected trader at the selected location.

4. Changing the Subcanvas

It's occasionally useful to use a different Canvas for each major (or minor) tab. For example, rather than having multiple overlapping windows, the Notebook widget can be used to provide many interfaces in one location. (However, this prevents the user from viewing two interfaces simultaneously!)

Ex 7. File–in Notebook4Example from the `online` examples directory. Browse the class and experiment with the tabs. Which method does all the work?