Chapter 1

Plugins in Nautilus

1.1 Plugin Manager

Nautilus provides several plugins and, of course, you can build your own. To see the available list of plugins and add or remove them, click in the top right arrow of a Nautilus browser and select Nautilus Plugin Manager.

Once you selected a plugin, it will be active the next time you open a new browser.

How to do enable

1.2 Some cool predefined plugins

AnnotationPane.

This plugin shows the meta information of methods such as author, last modification and others. Figure 1.2 shows an example.

Tasks.

This plugin shows the methods with a flag: within the package. This is really handy to monitor what you should do next. Figure 1.2 shows an example. When you click on the list of flagged elements you jump to the element.

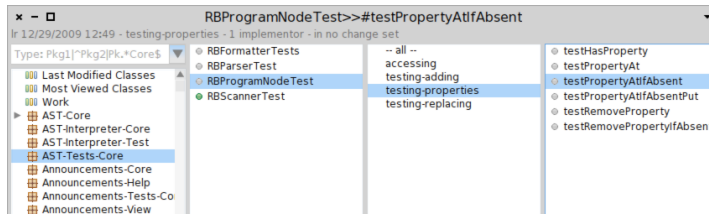

Figure 1.1: Method Annotation.

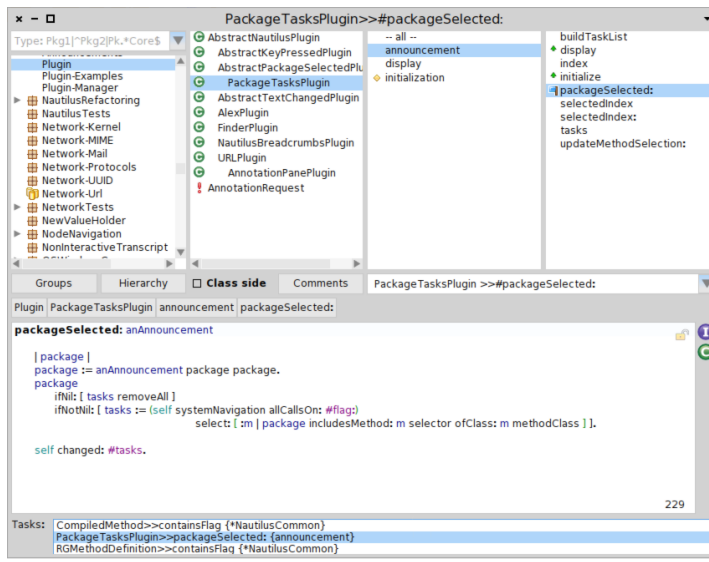

Figure 1.2: Package Tasks.

BreadCrumbs. This plugin offers a navigation path to the currently selected element. All the menu are available on the elements.

Figure 1.3 shows an example.

1.3 How to create Nautilus-Plugins

Here we will give some brief explanations on how to create your own plugin. There are only two requirements to create a Nautilus-Plugin:

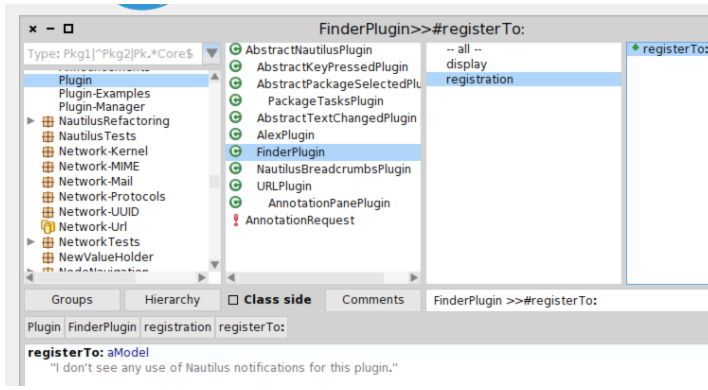

Figure 1.3: Navigable path (BreadScrums).

- the class should inherit from AbstractNautilusPlugin
- it should implement the method registerTo: aModel

```
AbstractNautilusPlugin
```

Announcement subscription

The method registerTo: is used by the plugin to register itself to aModel announcements.

```
MyPlugin>>registerTo: aModel

    aModel announcer
     when: NautilusKeyPressed
     send: #keyPressed:
     to: self
```

In this example, the instance of MyPlugin subscribes itself to NautilusKeyPressed, and tell aModel's announcer to send the message keyPressed to the instance.

So each time a key will be pressed in a Nautilus window the method keyPressed: will be called.

Display

If you want your plugin to add a graphical widget to Nautilus you should override the display method. This method should return the Morphic ele-

ment you want Nautilus to display. By default the method returns nil to notify Nautilus not to display anything.

```
MyPlugin>>display
   morph := LabelMorph new contents: 'MyPlugin';
      enabled: false;
      vResizing: #shrinkWrap;
      hResizing: #spaceFill;
      yourself.
   ^ morph
```

You can also redefine the following method on the class side:

- defaultPosition defines the default position of the morph. Possible values are {#top, #middle, #bottom, #none}. The default value is #none.

- possiblePositions answers a collection of the possible positions the widget can adopt.

Describing your plugin

And finally you can redefine the pluginName method to change the name displayed in the Nautilus Plugin Manager.

```
MyPlugin class>>description
   ^ 'MyPlugin'
```

```
MyPlugin class>>description
   ^ 'A super cool plugin'
```

Check in the Plugin Manager

Now your plugin should appear in the plugin list of the plugin manager and in Nautilus once you select it.