Copyright ©1996 by NeXT Software, Inc. All Rights Reserved.

WMInspector

Inherits From: Object

Declared In: apps/Workspace.h

Class Description

The WMInspector class defines the link between the Workspace Manager application and the module that's loaded into the application. When you build a new inspector for the Workspace Manager, you must create a subclass of WMInspector. The inspector you define must load its interface (that is, the nib file containing the interface) in its **new** method. It must also override the inherited **revert:** method to load information about the selection into its display.

Your inspector can query the Workspace Manager for information on the selection in the File Viewer by sending itself **selectionCount** and **selectionPathInto:separator:** messages. It can send itself **okButton**, **revertButton**, and **window** messages to gain access to those features of the Inspector panel.

Although the Contents inspector's principal role is to let the user view the contents of a File€Viewer entry, it can also let the user edit the displayed data. It's best not to overuse this€capability, however, since the Contents inspector wasn't designed to substitute for normal applications.

An inspector that permits editing should send itself a **touch:** message when the user begins modifying the data. This message enables the inspector's OK and Revert buttons and displays a broken ªXº in the panel's close box. (See **textDidChange:** for an alternate way to achieve this result.) The inspector should implement the **ok:** method to commit the modifications the user has made.

Instance Variables

id **window**;
id **okButton**;
id **revertButton**;
id **dirNameField**;
id **dirTitleField**;
id **fileNameField**;
id **fileIconButton**;

window	The Inspector window.
okButton	The Inspector's OK button.
revertButton	The Inspector's Revert button.
dirNameField	The TextField that holds the current directory.
dirTitleField	The TextField that titles **dirNameField.**
fileNameField	The TextField that displays the file name.
fileIconButton	The Button that displays the file's icon.

Method Types

Accessing the inspector object	+€new
Accessing panel controls	-€okButton -€revertButton -€window
Accessing Workspace selection	-€selectionCount -€selectionPathsInto:separator:
Managing changes	-€ok: -€revert: -€textDidChange: -€touch:

Class Methods

new

+€**new**

Creates a new WMInspector if none exists, or returns the existing one. When the object is created, it must load the nib file that contains the inspector's display.

The Workspace Manager sends a **new** message whenever it needs to access the inspector object. Thus, your subclass of WMInspector should ensure that no more than one instance of its class is created:

```
static id ribInspector = nil;

+ new
{
    if (ribInspector == nil) {
        char path[MAXPATHLEN+1];
        NXBundle *bundle = [NXBundle bundleForClass:self];

        self = ribInspector = [super new];
        if ([bundle getPath:path
                forResource:"RIBInspector"
                ofType:"nib"]) {
            [NXApp loadNibFile:path owner:ribInspector];
        } else {
            fprintf (stderr, "Couldn't load RIBInspector.nib\n");
            ribInspector = nil;
        }
    }
    return ribInspector;
}
```

Instance Methods

ok:

-€**ok:***sender*

Implement in your subclass to commit the changes that the user has made to the selected item. The OK button in the Inspector panel sends an **ok:** message when the user clicks it.

This method is optional, but if you implement it, you must send the same message to **super** as part of your implementation:

```
ok:sender
{
    /* your code to commit changes */
    [super ok:sender];
    return self;
```

```
}
```

This message to **super** replaces the broken ªXº in the panel's close box with the standard ªXº, indicating that the changes have been saved.

See also: -€**revert:**, -€**touch:**

okButton

-€**okButton**

Returns the **id** of the Inspector's OK button. This can be useful if you want to alter its title, for example.

See also: -€**revertButton:**

revert:

-€**revert:***sender*

Implement in your subclass to load data into the inspector's display. The Workspace Manager sends this message to the inspector object whenever the inspector's display might need to be updated; for example, when the Inspector panel is opened or when the selection changes in the File Viewer.

Your subclass must implement this method, and it must send the same message to **super** as part of its implementation:

```
revert:sender
{
    /* your code to show contents of selected item(s) */
    [super revert:sender];
    return self;
}
```

This message to **super** replaces the broken ªXº in the panel's close box with the standard ªXº, indicating that the changes have been discarded.

See also: -€**ok:**, -€**touch:**

revertButton

-€**revertButton**

Returns the **id** of the Inspector's Revert button. This can be useful if you want to alter its title, for example.

See also: -€**okButton:**

selectionCount

-€(unsigned)**selectionCount**

Returns the number of items selected in the File Viewer. You can use this information to determine whether your inspector should be displayed. For example, most inspectors can give information on only one file at a time, so within their **revert:** methods, they would have this test:

```
if ([self selectionCount] != 1) {
    return nil;
} else {
    /* get the path and display the file's contents */
}
```

See also: -€**selectionPathsInto:separator:**

selectionPathsInto:separator:

-€selectionPathsInto:(char *)*pathString* **separator:**(char)*character*

Returns the paths of the files selected in the File Viewer. The paths are place in the string *pathString*; each path is separated from the previous one by *character*. For example, if *character* is `:', *pathString* could contain ª/me/test1:/me/test2:/me/test3º.

If your inspector acts on only one file at a time (see **selectionCount**), the file's path can be identified using this message:

```
char fullPath[MAXPATHLEN+1];
[self selectionPathsInto:fullPath separator:'\0'];
```

See also: **-€selectionCount**

textDidChange:

-€textDidChange:*sender*

Sends the WMInspector a **touch:** message on behalf of some Text object in the Inspector panel.

By making your inspector object the delegate of any Text object in your inspector's display, the Inspector panel will be updated appropriately as the user alters the panel's contents.

See also: **-€touch:**

touch:

-€touch:*sender*

Changes the image in the Inspector panel's close box to a broken ªXº to indicate that the contents has been edited. Also, enables the OK and Revert buttons.

See also: **-€textDidChange:**

window

-€window

Returns the **id** of the window that contains the user interface for the inspector.