MiscInfoController

Inherits From: Object

Declared In: <misckit/MiscInfoController.h>

Class Description

The MiscInfoController sits underneath the Info submenu in the user interface and implements all the target-action methods sent by the Info submenu. The following menu items are supported by default, but more could always be added if necessary: Info..., License..., Release Notes..., Send Suggestion..., Show Menus..., Registration..., and Order Form... (When the Misc Preferences system is completed, Preferences... will also be supported.)

The Info..., Registration..., and Order Form... options use instances of the MiscInfo, MiscRegistration, and MiscOrderForm classes, respectively, to implement their functionality. The target/action messages to bring out these panels are simply forwarded on to the appropriate helper object, so you should check the documentation for these classes for more information on the options available.

The License... and Release Notes... both bring up the NXHelpPanel with specific pages loaded in the panel,

either the license (ªLicense.rtfdº) or the release notes (ªReleaseNotes.rtfdº), respectively. If you need to change the path names to these filese, you can do so with the ±**setLicenseFile:** and ±**setReleaseNotesFile:** methods.

You can use the ±**hideOrShowMenus:** method to show all the application's menus. Sending the action again will hide all the menus. This method is designed to be called by a MenuCell.

The Send Suggestion... item brings up a Compose window in Mail.app for the user to fill out and send to the author of the application (or, rather, the support e-mail address for the application). The message is addressed to an arbitrary support address and the subject line contains the name of the application, the serial number (if applicable), and the version number of the application. The body of the message is also arbitrary.

In order to provide an email address and the body of the message, a NXStringTable is used. If an NXStringTable is not connected up in the MiscInfoController's .nib file, the MiscInfoController will look for a string table in the appropriate .lproj directory (not yet implemented) or try to obtain a string table from the NXApp application object or it's delegate, in that order. The string table should contain the following keys:

AuthorEMail	Ð	the e-mail address to which the mail message should be sent
AuthorName	Ð	the name of the person to whom the suggestion mail message should be addressed
MailMessage	Ð	the body of the mail message sent as feedback; it should be a C format string with the key %s inserted twice, first for the addressee of the mail message and second for the name of the application. For example: ª%s,\n I really like %s but wish that it could also:\n\nº

When using the MiscInfoController in an application, you should create an instance of MiscInfoController in your main .nib file and connect the Info menu items to it. You can optionally connect up instances of MiscInfo, MiscRegistration, and MiscOrderForm objects (or subclasses) to the MiscInfo object. If you do not, the MiscInfoController will create them when necessary and set up all the connections which are necessary. See the documentation for each class for further information about how to set up each specific panel. Also check out the already connected example .nib files supplied with the MiscKit examples.

Instance Variables

id **strings**;
id **info**;
id **orderForm**;
id **registration**;

strings	String table to hold strings used by the UI and Send Suggestion... command.
info	A MiscInfo instance, which manages the Info... Panel.
orderForm	A MiscOrderForm instance, which manages the Order Form... Panel.
registration	A MiscRegistration instance, which manages the Registration... Panel.

Method Types

Accessing Helper Objects	- info - orderForm - registration - strings
Target/Action Methods	- hideOrShowMenus: ± info: - license: - orderForm: - releaseNotes: - registration:

	- suggestion:
Help file names	- licenseFile ± releaseNotesFile - setLicenseFile: - setReleaseNotesFile:

Instance Methods

hideOrShowMenus

- **hideOrShowMenus:***sender*

Uses the MiscExplodingMenu class to show or hide all the application's menus.

info

- **info**

Returns an instance of MiscInfo, creating it if necessary.

See also: ±info:

info:

- **info:***sender*

Uses the MiscInfo class to bring up an Info panel. Returns **self**.

See also: ±info

license:

- **license:***sender*

Brings up the NXHelpPanel displaying the file ªLicense.rtfdº. Returns **self**.

See also: ± licenseFile and **± setLicenseFile:**

licenseFile

- **licenseFile**

Returns a MiscString containing the name of the file opened by the ±**license:** method. You are responsible for freeing the string, as a copy is returned to you.

See also: ± license: and **± setLicenseFile:**

orderForm

- **orderForm**

Returns an instance of MiscOrderForm, creating it if necessary. If a custom subclass of MiscOrderForm is to be used, and this should usually be the case, then an instance of the custom MiscOrderForm subclass should be connected up to the MiscInfoController in the .nib file in which theMiscInfoController resides, which is most likely the main .nib file for the application. If this is not done, a MiscOrderForm will be used rather than the custom subclass.

See also: ±orderForm:

orderForm:

- **orderForm:***sender*

Brings up an Order Form panel which can be filled in and then printed in order to order copies of the application. The MiscOrderForm class will need to be subclassed in order for this feature to be very useful. Returns **self**.

See also: ±orderForm

registration
- **registration**

Returns an instance of MiscRegistration, creating it if necessary.

See also: ±registration:

registration:
- **registration:***sender*

Brings up a modal Registration panel in which the user is expected to enter a valid registration key. Returns **self**.

See also: ±registration

releaseNotes:
- **releaseNotes:***sender*

Brings up the NXHelpPanel displaying the file ªReleaseNotes.rtfdº. Returns **self**.

See also: ± releaseNotesFile and **± setReleaseNotesFile:**

releaseNotesFile

- **releaseNotesFile**

Returns a MiscString containing the name of the file opened by the ±**license:** method. You are responsible for freeing the string, as a copy is returned to you.

See also: ± releaseNotes: and **± setReleaseNotesFile:**

setLicenseFile

- **setLicenseFile:**(MiscString *)*aPath*

Takes the stringValue of *aPath* and assigns internally for use as the pathname of the help page to be brought up upon invocation of the **license:** method. This path name should start with a ª/º character and will be appended to the path of the Help file directory. The default is ª/License.rtfdº, which by the convention explained here means the file ª/pathToTheApp/xxx.app/YourLanguage.lproj/Help/License.rtfdº (with the path to the app, the appname, and the proper language project as set up on the system on which the app is running.

See also: ± license: and **± licenseFile**

setReleaseNotesFile

- **setReleaseNotesFile:**(MiscString *)*aPath*

Takes the stringValue of *aPath* and assigns internally for use as the pathname of the help page to be brought up upon invocation of the **releaseNotes:** method. This path name should start with a ª/º character and will be appended to the path of the Help file directory. The default is ª/ReleaseNotes.rtfdº, which by the convention explained here means the file ª/pathToTheApp/xxx.app/YourLanguage.lproj/Help/ReleaseNotes.rtfdº (with the path to the app, the appname, and the proper language project as set up on the system on which the app is running.

See also: **± releaseNotes:** and **± releaseNotesFile**

strings

- **strings**

Returns the MiscInfoController's string table. If one has not already been set (via connections in the .nib file), then the .lproj directories will be searched for ªMiscInfoController.stringsº (not yet implemented). If the file is not found, then first the application, and then its delegate object, will be checked to see if they have a string table available, using the ±**strings** method. If neither responds to this message, then the MiscInfoController will give up and return *nil*.

suggestion:

- **suggestion:***sender*

Brings up a Compose window in Mail.app, first presenting an appropriate warning to avoid clobbering any currently open Compose windows. The Compose window is filled with a boiler plate mail message addressed to the author of the application and containing the version number and registration status (serial number if registered) of the application. Returns **self**.