MiscDocument

Inherits From:	MiscNibController : Object
Declared In:	MiscKit/MiscDocument.h

Class Description

MiscDocument is a subclass of MiscNibController.   Along with MiscDocManager it provides a complete framework for document based applications.   MiscDocument extends MiscNibController by providing file support.   The window which MiscDocument inherits from MiscNibController can be attached to a file.   MiscDocument provides the framework for reading and writing document files.   All the standard Document menu items are supported.   Multiple file types are supported.   A MiscDocument subclass can support reading and writing of any number of file types with different extensions (or no extension). Along with MiscDocManager many more features are supported including untitled document numbering, window staggering, and full document menu management.

MiscDocument is an abstract superclass. You must subclass it to use it. In your subclass' **+initialize** method, you must configure certain things. The class nib file should be set with the MiscNibController method **+setClassNib:**. The overall name for documents of this type should be set using **+setControllerName:**. Then, for each type of file that the class can read, write, or read and write, **+addDocType:name:extension:openSelector:saveSelector:**. The first argument is a numerical tag which is purely for your use. The name is the full name of this type of document file (ie "Rich Text"). The extension is the filename extension used for files of this type (ie "rtf"). openSelector is the method to call to read files of this type (or NULL if this type can't be read). saveSelector is the method to call to save files of this type (or NULL if files of this type can't be saved). Each type should have at least one of openSelector and saveSelector non-NULL. Otherwise it's a useless type. Open and save selectors both take one argument, the MiscDocType object representing the file type to be opened or saved. Note that it is possible to have separate methods to read and write each type you support, or you can have one to read all types and one to write all types. In the latter case, the argument comes in handy.

Each subclass of MiscDocument must implement several methods. Two have fixed identities: **-doClear** and **-doPrint**. The first should initialize the document object to an empty state (as if the user just said "New"). This is used in two cases. First, it is called to set up new documents when they are created. Also, it is used if the user chooses "Revert" before the file has been saved. Therefore this method must be able to completely recycle the document, even if it has contents already. **-doPrint** is used to print the document. It should run the print panel first.

The other methods that MiscDocument subclass' need to implement are those that were given as open or save selectors with **+addDocType:name:extension:openSelector:saveSelector:** calls. You must implement each method you named as an open or save selector. These methods take a single object (of class MiscDocType) as an argument. The MiscDocType represents the file type meant to be read or saved.

The document type information is used in several ways. It is used internally by the **-saveAs:** method to fill a popup menu that is put in the save panel if the MiscDocument supports multiple save types. The menu allows users to choose what type they want the file to be. Type information is also used by MiscDocManager when it runs the open panel to allow the user to open only those files that the MiscDocument supports.

One more thing your subclass should do is make sure that **-setDirty:** method is called whenever the document changes such that it will need to be saved.

To create MiscDocument instances, you normally use the **-initFromFile:** method. A NULL filename means a new untitled document. Usually, MiscDocManager will call this method, and most of the others in MiscDocument. See the MiscDocManager class docs for details.

It is possible to use a MiscDocument without a manager. It is also possible to have a MiscDocument without a window. This might be particularly useful in implementing a MiscDocument that had no window, but managed a set of document view objects in an application that allowed multiple views on one document. Having said that, I should say that the MiscDocument has not been well tested operating with a manager or window.

See the DocArchitecture example to see how MiscDocManager and MiscDocument are used.

Instance Variables

```
MiscDocManager     *manager;
MiscString         *docName;
MiscString         *untitledString;
MiscDocType        *docType;
BOOL               isDirty;
id                 saveAccessoryPanel;
id                 saveAccessoryBox;
id                 saveAccessoryPopupButton;
```

manager	The MiscDocManager for this controller.
docName	This is always a valid MiscString object. If the document has a file attached to it, this is the pathname. If the doc is still untitled, it's empty

unititledString	This is always a valid MiscString object. If the doc is untitled, this string is used to hold the window title.
docType	If the document is untitled, this is nil. Otherwise it points at the MiscDocType object that represents the document type of the file attached to the document.
isDirty	True if the document needs saving.
saveAccessoryPanel	This is an IB outlet. It points at the panel containing the default save panel accessory to use for this class. The default save accessory is a box with a popup menu inside. (See the next two instance variables.)
saveAccessoryBox	This is an IB outlet. This points at the actual default save accessory.
saveAccessoryPopupButton	This is an IB outlet. This points at the button fronting for the popup menu of save types.

Method Types

Initializing the class	+ initialize + startUnloading

Class controller name	+ setControllerName: + controllerName
Class document types	+ addDocType:name:extension:openSelector:saveSelector: + docTypes + docTypeForTag: + docTypeForName: + docTypeForExtension: + getOpenTypesList: + getSaveTypesList:
Initializing instances	- initFromFile:manager: - initWithFrameName: - initWithFrameName:fromFile:manager: - free
	Loading the nib - nibDidLoad
Subclass responsibilities	- doClear - doPrint
Save accessory support	- saveAccessoryPopupAction: - getSaveAccessoryForSaveTypes:currentType:

Document menu support	- save:
	- saveAs:
	- saveTo:
	- revert:
	- close:
	- print:
	- saveIfNeeded
The manager	- setManager:
	- manager
The file	- setFile:
	- isUntitled
	- file
	- docName
Window support	- resetWindowTitle
	- setDirty:
	- isDirty
	- windowDidBecomeMain:
	- windowDidResignMain:
	- windowWillClose:

Archiving	- awake
	- read:
	- write:

Class Methods

addDocType:name:extension:openSelector:saveSelector:

+ **addDocType:**(int)*tag*
    name:(const char *)*name*
    extension:(const char *)*ext*
    openSelector:(SEL)*openSel*
    saveSelector:(SEL)*saveSel*

Adds a new doc type to the list of document types maintained by the class. The list is kept in a MiscClassVariable, so separate lists are kept for any subclasses of MiscDocument. Document types are storted in MiscDocType objects. The tag is purely for the subclasser's use. Name should be the full name of the document type (ie "Rich Text"). Extension should be the file name extension to use (ie "rtf"). A null extension means no extension (ala Edit text files). The open ans save selectors should be the methods to use to open and save documents of this type. Either can be NULL if reading or writing a particular type is not supported. The methods should take one argument, the MiscDocType object which represents the type to open or save.

See also: + **docTypes**, + **docTypeForTag:**, + **docTypeForName:**, + **docTypeForExtension:**, + **getOpenTypesList:**, + **getSaveTypesList:**

controllerName

+ (MiscString *)**controllerName**

Sets the general name for documents of this class. MiscDocManager uses this name when constructing New and Open submenus if it is managing more than one document class.

See also: + **setControllerName:**

docTypeForExtension:

+ (MiscDocType *)**docTypeForExtension:**(const char *)*extension*

Returns the first document type in this class' document type list with the given extension. Document types should probably have unique extensions within a subclass of MiscDocument.

See also: + **addDocType:name:extension:openSelector:saveSelector:**, + **docTypes**, + **docTypeForTag:**,

+ **docTypeForName:**, + **getOpenTypesList:**, + **getSaveTypesList:**

docTypeForName:
+ (MiscDocType *)**docTypeForName:**(const char *)*name*

Returns the first document type in this class' document type list with the given name. Document types should probably have unique names within a subclass of MiscDocument.

See also: + **addDocType:name:extension:openSelector:saveSelector:**, + **docTypes**, + **docTypeForTag:**, + **docTypeForExtension:**, + **getOpenTypesList:**, + **getSaveTypesList:**

docTypeForTag:
+ (MiscDocType *)**docTypeForTag:**(int)*tag*

Returns the first document type in this class' document type list with the given tag. Document types should probably have unique tags within a subclass of MiscDocument.

See also: + **addDocType:name:extension:openSelector:saveSelector:**, + **docTypes**, + **docTypeForName:**, + **docTypeForExtension:**, + **getOpenTypesList:**, + **getSaveTypesList:**

docTypes

+ (List *)**docTypes**

Returns the list of document types for this class.

See also: + **addDocType:name:extension:openSelector:saveSelector:**, + **docTypeForTag:**, + **docTypeForName:**, + **docTypeForExtension:**, + **getOpenTypesList:**, + **getSaveTypesList:**

getOpenTypesList:

+ (List *)**getOpenTypesList:**(List *)*list*

Fills *list* with all the document types for this class which have a non-NULL openSelector. If *list* is nil, a List object is allocated for it. The list is returned.

See also: + **addDocType:name:extension:openSelector:saveSelector:**, + **docTypes**, + **docTypeForTag:**, + **docTypeForName:**, + **docTypeForExtension:**, + **getSaveTypesList:**

getSaveTypesList:

+ (List *)**getSaveTypesList:**(List *)*list*

Fills *list* with all the document types for this class which have a non-NULL saveSelector. If *list* is nil, a List object is allocated for it. The list is returned.

See also: + **addDocType:name:extension:openSelector:saveSelector:**, + **docTypes**, + **docTypeForTag:**, + **docTypeForName:**, + **docTypeForExtension:**, + **getOpenTypesList:**

initialize

+ **initialize**

Sets the class' version. Loads all necessary classes. Initializes the class' class variables.

See also: + **startUnloading**

setControllerName:

+ **setControllerName:**(const char *)*typeName*

Sets the general name for documents of this class. MiscDocManager uses this name when constructing New and Open submenus if it is managing more than one document class.

See also: + **controllerName**

startUnloading
  + **startUnloading**

Frees the class variables.

See also: + **initialize**

Instance Methods

awake
  - **awake**

Initializes instance variables that aren't written to typed stream.

See also: ± **read:**, ± **write:**

close:
  - **close:***sender*

Closes the document window.

See also: ± **save:**, ± **saveAs:**, ± **saveTo:**, ± **revert:**, ± **print:**, ± **saveIfNeeded**, ± **windowWillClose:**

doClear

- (BOOL)**doClear**

This method should be implemented by all subclasses. It should be able to empty an existing document and leave it in the state of a brand new, ready to use document. This is called to initialize new documents, and to revert untitled documents.

See also: ± **doPrint**

doPrint

- (BOOL)**doPrint**

This method should be implemented by all subclasses. It should print the document, running the print panel first.

See also: ± **doClear**

docName

- **docName**

Returns the MiscString instance variable docName.   Usually the file method should be used to find out the document's name, but if you need access to the MiscString object (for instance to peel off a file extension, or something), this method can be used to access the MiscString itself.

See also: ± **setFile:**, ± **isUntitled**, ± **file**

file

- (const char *)**file**

Returns the name of the document.   If the doc is untitled, the untitledString's contents are returned, otherwise, the docName's contents are returned.

See also: ± **setFile:**, ± **isUntitled**, ± **docName**

free

- **free**

Frees the docName and untitledString.   Also removes us from our manager if we have one.

See also: ± **initFromFile:manager:**, ± **initWithFrameName:**, ± **initWithFrameName:fromFile:manager:**

getSaveAccessoryForSaveTypes:currentType:

- **getSaveAccessoryForSaveTypes:**(List *)*saveTypes*
  currentType:(MiscDocType *)*currentType*

Loads the saveAccessory nib file, if necessary, then, if this document class supports mulitple save types, the saveAccessoryPopupButton's popup menu is filled with the types the document class can save and the saveAccessoryBox is returned.   If the document class supports only one save type, nil is returned, and no save accessory is used.   You may completely override this method if you need other things in your save accesory. If you do so, use new instance variables for your IB outlets, don't use the existing saveAccessory... outlets. Also, if you do override this, make sure that the popup menu in your new save accessory has it's action set to **±saveAccessoryPopupAction:**.

See also: ± **saveAccessoryPopupAction:**

initFromFile:manager:

- **initFromFile:**(const char *)*path*

manager:*aManager*

Initializes a new instance. If path is non-NULL, the file is attached to the document and read. If path is NULL, the document starts out untitled and empty. This is the recommended way of initializing instances. Most of the time this method will be called by a MiscDocManager.

See also: ± **initWithFrameName:**, ± **initWithFrameName:fromFile:manager:**, ± **free**

initWithFrameName:

- **initWithFrameName:**(const char *)*theFrameName*

Calls **±initWithFrameName:**framename **fromFile:**NULL **manager:**nil.

See also: ± **initFromFile:manager:**, ± **initWithFrameName:fromFile:manager:**, ± **free**

initWithFrameName:fromFile:manager:

- **initWithFrameName:**(const char *)*theFrameName*
  fromFile:(const char *)*path*
  manager:*aManager*

This is the designated initializer for the class, but most of the time, the **±initFromFile:manager:** method is preferrable.

See also: ± **initFromFile:manager:**, ± **initWithFrameName:**, ± **free**

isDirty

- (BOOL)**isDirty**

Returns YES if the document has changed since it was last saved.   NO otherwise.

See also: ± **setDirty:**

isUntitled

- (BOOL)**isUntitled**

Returns YES if this document is untitled (ie its [docName isEmpty]).

See also: ± **setFile:**, ± **file**, ± **docName**

manager

- **manager**

Returns the MiscDocManager which manages this document.

See also: ± **setManager:**

nibDidLoad

- **nibDidLoad**

This method makes sure the window's title is correct, and if the document has a manger, it uses it to stagger the window location.

print:

- **print:***sender*

This method is usually called by MiscDocManager, but it is set up as an action method so that you can connect the Print menu item directyl to an instance, or to the FirstResponder. This prints the document, but note that all the work is shunted to the **±doPrint** method which subclasses must provide.

See also: ± **save:**, ± **saveAs:**, ± **saveTo:**, ± **revert:**, ± **close:**, ± **saveIfNeeded**

read:

- **read:**(NXTypedStream *)*strm*

Reads the controller from typed stream.   DO NOT use read and write to save your documents.   In fact, I don't know why I even provide this method as it makes little sense.

See also: ± **awake**, ± **write:**

resetWindowTitle

- **resetWindowTitle**

Sets the window's title to something appropriate (ie the file's anme, or the untitled string).

revert:

- **revert:***senderender*

Revert's the document to its last saved state, or back to empty if it's untitled.   This is usually called by MiscDocManager.   The work is shunted to either **±doClear** (for untitled docs) or the openSelector for the current document type if the doc has a file.

See also: ± **save:**, ± **saveAs:**, ± **saveTo:**, ± **close:**, ± **print:**, ± **saveIfNeeded**

save:

- **save:***sender*

Saves the docuemnt. This is usually called by MiscDocManager. The actual work is done by the current doc type's saveSelector. If the doc is untitled, this calls **±saveAs:** instead.

See also: ± **saveAs:**, ± **saveTo:**, ± **revert:**, ± **close:**, ± **print:**, ± **saveIfNeeded**

saveAccessoryPopupAction:

- **saveAccessoryPopupAction:***senderç„*

The popup menu in the save accessory has this as its action. It sets the save panel's "Required file type" to whatever type was chosen from the menu.

See also: ± **getSaveAccessoryForSaveTypes:currentType:**

saveAs:

- **saveAs:***sender*

Runs the save panel to get a name (and doc type) for the doc, then calls **±save:**.   This is usually called by MiscDocManager.

See also: ± **save:**, ± **saveTo:**, ± **revert:**, ± **close:**, ± **print:**, ± **saveIfNeeded**

saveIfNeeded

  -   **saveIfNeeded**

If the document is dirty, this calls **±save:**.   Otherwise, it just returns.

See also: ± **save:**, ± **saveAs:**, ± **saveTo:**, ± **revert:**, ± **close:**, ± **print:**

saveTo:

  -   **saveTo:***sender*

This runs the save panel to get a new file name (and doc type), saves the document into that file with **±save:**, then sets the file anme back to the old name.

See also: ± **save:**, ± **saveAs:**, ± **revert:**, ± **close:**, ± **print:**, ± **saveIfNeeded**

setDirty:

- **setDirty:**(BOOL)*flag*

Sets the dirtiness of the document. Subclasses should insure that **±setDirty:**YES is called whenever a change is made in the document.

See also: ± **isDirty**

setFile:

- **setFile:**(const char *)*fName*

Generally you won't call this method. It is used by several internal methods to set the file attached to a document.

See also: ± **isUntitled**, ± **file**, ± **docName**

setManager:

- **setManager:***aManager*

You won't generally call this method. Usually, the manager for a doc is set when it's initialized. This can be used to reset the manager.

See also: ± **manager**

windowDidBecomeMain:

- **windowDidBecomeMain:***sender*

This is used to inform our MiscDocManager that we are now the current document.

See also: ± **windowDidResignMain:**

windowDidResignMain:

- **windowDidResignMain:***sender*

This is used to inform our MiscDocManager that should no longer be the current document.

See also: ± **windowDidBecomeMain:**

windowWillClose:

- **windowWillClose:***sender*

This method gives the user a chance to save if the document needs saving. Then, if the user doesn't cancel the close, the window is closed, and the MiscDocument is freed (through NXApp's **±delayedFree:** method.

See also: ± **close:**

write:

- **write:**(NXTypedStream *)*strm*

Writes the controller to a typed stream. DO NOT use read and write to save your documents. In fact, I don't know why I even provide this method as it makes little sense.

See also: ± **awake**, ± **read:**