MODocManager

Inherits From: Object

Declared In: MOKit/MODocManager.h

Class Description

MODocManager manages a bunch of instances of one or more MODocController subclass.   It provides support for managing the Document menu, numbering untitled documents, staggering window positions, and giving the user a chance to save things before quitting.

An application will generally have one MODocManager.   The Document menu's items will have their actions connected to the various appropriate MODocManager methods.   The MODocManager should be the application delegate or you should make sure that **±appWillTerminate:** messages at least get passed on to it. MODocManagers menu outlets should be connected to the appropriate Document menu items.

To configure a MODocManager you must also give it the set of document classes it will manage.   Most document based applications will have a single document type.   MODocManager also supports applications which require multiple document types.   Use the **±addDocumentClass:** method to inform the manager about each MODocController subclass you intend to use documents from.   If there is one document class, the Document menu is left alone and will be standard.   If there is more than one, the New item is replaced by a New submenu which has items for each class.   The text used for the items is taken from MODocController's **+controllerName** method.   Also, in addition to the normal Open item, an Open submenu is added with items for each class just like the New submenu.   The Open item allows opening documents of any type supported by any of the document classes.   The submenu lets you be specific.

MODocManager, with the cooperation of MODocController, keeps a list of all the currently open documents and keeps track of the current document.   Document menu actions which act on a specific document, act on the current document, which is the document to which the main window belongs.

MODocController also uses MODocManager to number its untitled documents, and to stagger its windows so they don't stack up.   Also, when the user quits, if there are any unsaved documents, the user is given a chance to save first.

See the DocArchitecture example to see how MODocManager and MODocController are used.

Instance Variables

List	***docList**;
MODocController	***currentDoc**;
List	***docClassList**;
int	**untitledCount**;
int	**frameCycle**;
NXPoint	**windowStartingLocation**;
id	**quitPanel**;
id	**buttonMatrix**;
id	**docMenu**;
id	**openCell**;
id	**newCell**;

id	**saveCell**;
id	**saveAsCell**;
id	**saveToCell**;
id	**saveAllCell**;
id	**revertCell**;
id	**closeCell**;
Menu	***newMenu**;
MenuCell	***newSubmenuCell**;
Menu	***openMenu**;
MenuCell	***openSubmenuCell**;

docList	A List object containing all the document controller instances currently being managed.
currentDoc	The document controller instance to which the current main window of the app belongs.
docClassList	A List object containing all the document controller classes that we are managing instances for.

untitledCount	Counter to number untitled windows.
frameCycle	Counter to stagger window locations.
windowStartingLocation	The base location to start the first opened document window at. Subsequent windows are staggered from there.
quitPanel	IB outlet for the quit panel which gives the user a chance to save things before quitting.
buttonMatrix	IB outlet. Some buttons from the quit panel
docMenu	IB outlet which should point to the Document submenu.
openCell	IB outlet which should point to the Open item in the Document Submenu.
newCell	IB outlet which should point to the New item in the Document Submenu.
saveCell	IB outlet which should point to the Save item in the Document Submenu.
saveAsCell	IB outlet which should point to the Save As item in the Document Submenu.
saveToCell	IB outlet which should point to the Save To item in the Document Submenu.

saveAllCell	IB outlet which should point to the Save All item in the Document Submenu.
revertCell	IB outlet which should point to the Revert item in the Document Submenu.
closeCell	IB outlet which should point to the Close item in the Document Submenu.
newMenu	This points to the New submenu which is used if there is more than one document controller class being managed.
newSubmenuCell	This points to the New submenu item which is inserted into the Document menu if there is more than one document controller class being managed.
openMenu	This points to the Open submenu which is used if there is more than one document controller class being managed.
openSubmenuCell	This points to the Open submenu item which is inserted into the Document menu if there is more than one document controller class being managed.

Method Types

Initializing the class	+ initialize	
Initializing instances	± init	
	± initDocumentClass:	
	± free	
	± awakeFromNib	
	Loading the nib	± nibDidLoad
Document menu support	± new:	
	± newFromDocumentClass:	
	± open:	
	± openFromDocumentClass:	
	± openDocument:withClass:	
	± save:	
	± saveAs:	
	± saveTo:	
	± revert:	
	± close:	
	± print:	

	± saveAll:
MODocController support	± getNextWindowLocation:
	± nextUntitledNum
	± setStartingWindowLocation:
Document controller classes	± addDocumentClass:
	± removeDocumentClass:
	± documentClassList
	± docClassCount
Document controller instances	± addDocument:
	± removeDocument:
	± findDocumentForWindow:
	± documentList
	± makeDocumentsPerform:with:
	± areDocumentsDirty:
	± setCurrentDocument:
	± currentDocument
Save on quit support	± appWillTerminate:

	± quitPanelStopModalAction:
	± windowDidBecomeKey:
	± windowDidResignKey:
Menu management	± menuUpdate:
Archiving	± awake
	± read:
	± write:

Class Methods

initialize
+ **initialize**

Sets the class' version.   Loads all necessary classes.

Instance Methods

addDocument:

- **addDocument:**(MODocController *)*aDocument*

Adds a document controller instance to the list of documents being managed. This is called automatically when a MODocController subclass instance's **±setManager:** method is called or when it is initialized.

See also: ± **removeDocument:**, ± **findDocumentForWindow:**, ± **documentList**, ± **makeDocumentsPerform:with:**, ± **areDocumentsDirty**, ± **setCurrentDocument:**, ± **currentDocument**

addDocumentClass:

- **addDocumentClass:**(Class)*docControllerClass*

This method can be used to add to the list of MODocController subclasses that the manager knows about. The MODocManager needs to know about the various classes that are used for documents to properly manage the Document menu and run the open panel. When you call this method, the Document menu is rebuilt. If there is only one document class, the Document menu looks like normal (it has one Open item and a New item). If the manager is managing documents from more than one class, the menu changes. Instead of New being an item, it becomes a submenu with one entry for each class. The text used for the submenu is retrieved from the document class with the **+controllerName** method. Also, in addition to the normal Open menu item, an Open submenu is added beneath with an item for each controller class. The Open item will open files of any type handled by any document class. If more than one class supports reading the same file extension, the one first in the list is used to open the file. The Open and New submenus let you specify what type of document to create.

See also: ± **removeDocumentClass:**, ± **documentClassList**, ± **docClassCount**

appWillTerminate:

- **appWillTerminate:***sender*

If there are unsaved documents, this method gives the user a chance to save them. It asks via an alert panel whether to Quit qithout saving, save all before quitting, review each unsaved document individually, or cancel the quit.

See also: ± **quitPanelStopModalAction:**, ± **windowDidBecomeKey:**, ± **windowDidResignKey:**

areDocumentsDirty
- (BOOL)**areDocumentsDirty**

Returns YES if any of the documents being managed need saving. NO otherwise.

See also: ± **addDocument:**, ± **removeDocument:**, ± **findDocumentForWindow:**, ± **documentList**, ± **makeDocumentsPerform:with:**, ± **setCurrentDocument:**, ± **currentDocument**

awake
- **awake**

Initializes stuff after a **±read:** that doesn't get stored in the typed stream.

See also: ± **read:**, ± **write:**

awakeFromNib
- **awakeFromNib**

This sets the update action for all the document menu items if the IB outlets are set.   Then it rebuilds the menu to be in line with the document classes which have been registered through either **±initDocumentClass:** or **±addDocumentClass:**.

See also: ± **init**, ± **initDocumentClass:**, ± **free**

close:

- **close:***sender*

Closes the current document.   This should be the action for the Close item in the Document menu.

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

currentDocument

- **currentDocument**

Returns the current document (that is, the document to which the current main window belongs, or nil if there is no current document).

See also: ± **addDocument:**, ± **removeDocument:**, ± **findDocumentForWindow:**, ± **documentList**, ± **makeDocumentsPerform:with:**, ± **areDocumentsDirty**, ± **setCurrentDocument:**

docClassCount

- (int)**docClassCount**

Returns the number of MODocController subclasses that the manager knows about.

See also: ± **addDocumentClass:**, ± **removeDocumentClass**, ± **documentClassList**

documentClassList

- (List *)**documentClassList**

Returns the List object used to store all MODocController subclasses which this manager knows about. Don't change this list directly. Use **±addDocumentClass:** and **±removeDocumentClass:** to modify the list.

See also: ± **addDocumentClass:**, ± **removeDocumentClass**, ± **docClassCount**

documentList

- (List *)**documentList**

Returns the List object used to sotre all open documents. The items in the list are MODocController subclass instances. Do not modify this list.

See also: ± **addDocument:**, ± **removeDocument:**, ± **findDocumentForWindow:**, ± **makeDocumentsPerform:with:**, ± **areDocumentsDirty**, ± **setCurrentDocument:**, ± **currentDocument**

findDocumentForWindow:

- (MODocController *)**findDocumentForWindow:***aWindow*

Returns the MODocController which owns the given window. Returns nil if the window is not owned by a MODocController which is being managed by this MODocManager.

See also: ± **addDocument:**, ± **removeDocument:**, ± **documentList**, ± **makeDocumentsPerform:with:**, ± **areDocumentsDirty**, ± **setCurrentDocument:**, ± **currentDocument**

free

- **free**

Frees all the managed documents and the document list and the doc class list. Also restores the document menu to its original state and frees the New and Open submenus if they were ever created.

See also: ± **init**, ± **initDocumentClass:**, ± **awakeFromNib**

getNextWindowLocation:

- **getNextWindowLocation:**(NXPoint *)*pt*

Returns the location for the top-left corner of the next document window to open. MODocControllers which have managers use this method to position their windows. The location starts at a given point and then, for each document that opens it is staggered down and right. Once a certain number of windows have been opened, the location wraps back to the beginning. Basically this behaves like Edit, the windows march down the screen for a while, then, before they fall off the bottom, the cycle of locations starts to repeat.

See also: ± **nextUntitledWindowNum**, ± **setStartingWindowLocation:**

init

- **init**

Calls **±initDocumentClass:**nil.

See also: ± **initDocumentClass:**, ± **free**, ± **awakeFromNib**

initDocumentClass:

- **initDocumentClass:**(Class)*docControllerClass*

This is the designated initializer for this class. It sets up the instance with the given class as the only type of document managed. Without at least one document class, the New and Open menu items in the Document menu will never be enabled. More than one document class can be used. Add further ones with the **±addDocumentClass:** method.

See also: ± **init**, ± **free**, ± **awakeFromNib**

makeDocumentsPerform:with:

- **makeDocumentsPerform:**(SEL)*aMethod*
  with:*anArg*

Sends the given message to each open doicuemnt which is managed by this manager.

See also: ± **addDocument:**, ± **removeDocument:**, ± **findDocumentForWindow:**, ± **documentList**, ± **areDocumentsDirty**, ± **setCurrentDocument:**, ± **currentDocument**

menuUpdate:

- (BOOL)**menuUpdate:***menuCellldBecomeKey:sender*

This is the update action for all the Document menu items. Each item is enabled or disabled as appropriate for the current state of the App.

See also: ± **awakeFromNib**

new:

- **new:***sender*

This method is only used for managers which manage a single document class. It instantiates a new document of that class. This is the action of the New item in the Document menu. The New item is removed from the menu and replaced by a New submenu when the manager is managing more than one document class.

See also: ± **newFromDocumentClass:**, ± **open:**, ± **openFromDocumentClass:**, ± **openDocument:withClass:**

newFromDocumentClass:

- **newFromDocumentClass:***sender*

This method is used only when there is more than one document class. It instantiates an object of the class which appears at the same offset in the document class list as the selected row of the sender. Each item in the New submenu is connected to this action.

See also: ± **new:**, ± **open:**, ± **openFromDocumentClass:**, ± **openDocument:withClass:**

nextUntitledNum

- (int)**nextUntitledNum**

Returns the next unused untitled document number. MODocControllers which are managed call this to number untitled docs.

See also: ± **getNextWindowLocation**, ± **setStartingWindowLocation:**

open:

- **open:***sender*

This is the action of the Open item in the Document menu. It runs the open panel allowing selection of any file type supported for reading by any of the document classes this manager manages. If more than one document class can read the same file type, the one which appears first in the document class list is used to open files of that type. If there is more than one document class, files can be opened with a specific class by using the Open submenu and the **±openFromDocumentClass:** method.

See also: ± **new:**, ± **newFromDocumentClass:**, ± **openFromDocumentClass:**, ± **openDocument:withClass:**

openDocument:withClass:

- **openDocument:**(const char *)*path*
  withClass:(Class)*docClass*

Opens the named file with the given document class. Both **±open:** and **±openFromDocumentClass:** call this method after they have a filename from the open panel. Only call this method with combinations of path and docClass that can work together.

See also: ± **new:**, ± **newFromDocumentClass:**, ± **open:**, ± **openFromDocumentClass:**

openFromDocumentClass:

- **openFromDocumentClass:***sender*

This is the action of all the items in the Open submenu. It is only used when there is more than one document class being managed. It runs the Open panel allowing selection of all types supported for reading by the document class at the same offset in the document class list as the selected row of the sender.

See also: ± **new:**, ± **newFromDocumentClass:**, ± **open:**, ± **openDocument:withClass:**

print:

- **print:***sender*

This is the action of the Print item in the Document menu. It prints the current document.

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

quitPanelStopModalAction:

- **quitPanelStopModalAction:***sender*

This action is used while running the quit panel. It is the target of all the buttons. It just stops the modal session and sends the sender's tag back through the **±stopModal:** mechanism.

See also: ± **appWillTerminate:**, ± **windowDidBecomeKey:**, ± **windowDidResignKey:**

read:

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

Reads the object from a typed stream.   I don't know why I included this method since you probably won't be archiving these.
See also: ± **awake**, ± **write:**

removeDocument:

- **removeDocument:**(MODocController *)*aDocument*

Removes the given document from the document list.

See also: ± **addDocument:**, ± **findDocumentForWindow:**, ± **documentList**, ± **makeDocumentsPerform:with:**, ± **areDocumentsDirty**, ± **setCurrentDocument:**, ± **currentDocument**

removeDocumentClass:

- **removeDocumentClass:**(Class)*docControllerClass*

Removes the given document class from the document class list.   Then redoes the Document menu to reflect the new set of classes we're managing.

See also: ± **addDocumentClass:**, ± **documentClassList**, ± **docClassCount**

revert:

- **revert:***sender*

Action of the Revert item in the Document menu.   Reverts the current document.

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

save:

- **save:***sender*

Action of the Save item in the Document menu.   Saves the current document.

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

saveAll:

- **saveAll:***sender*

Action of the Save All item in the Document menu.   Saves all dirty open documents.

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

saveAs:

- **saveAs:***sender*

Action of the Save As item in the Document menu.   Does a Save As in the current document.

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

saveTo:

- **saveTo:***sender*

Action of the Save To item in the Document menu.   Does a Save To in the current document.

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

setCurrentDocument:

- **setCurrentDocument:**(MODocController *)*aDocument*

Sets the manager's current document.   Thgis is generally called from the **±windowDidBecomeMain:** and **±windowDidResignMain:** methods of MODocController.

See also: ± **addDocument:**, ± **removeDocument:**, ± **findDocumentForWindow:**, ± **documentList**, ± **makeDocumentsPerform:with:**, ± **areDocumentsDirty**, ± **currentDocument**

setWindowStartingLocation:

- **setWindowStartingLocation:**(const NXPoint *)*pt*

This method sets the location that the first window to open will appear at.   This is also the location used to compute the subsequent positions of other windows as they are opened.

See also: ± **getNextWindowLocation**, ± **nextUntitledNum**

windowDidBecomeKey:
- **windowDidBecomeKey:***sender*

This is for support of the quit panel. It puts the Return sign icon in the default button in the quit panel.

See also: ± **appWillTerminate:**, ± **quitPanelStopModalAction:**, ± **windowDidResignKey:**

windowDidResignKey:
- **windowDidResignKey:***sender*

This is for support of the quit panel. It removes the Return sign icon from the default button in the quit panel.

See also: ± **appWillTerminate:**, ± **quitPanelStopModalAction:**, ± **windowDidBecomeKey:**

write:

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

Writes the manager to a typed stream.   I don't know why I even provide this method as it makes little sense.

See also: ± **awake**, ± **read:**