Class

NSDocumentController

An object that manages an app's documents.

macOS

```
@MainActor
class NSDocumentController
```

Overview

As the first-responder target of New and Open menu commands, `NSDocumentController` creates and opens documents and tracks them throughout a session of the app. When opening documents, a document controller runs and manages the modal Open panel. `NSDocumentController` objects also maintain and manage the mappings of document types, extensions, and `NSDocument` subclasses as specified in the CFBundleDocumentTypes property loaded from the information property list (`Info.plist`).

You can use various `NSDocumentController` methods to get a list of the current documents, get the current document (which is the document whose window is currently key), get documents based on a given filename or window, and find out about a document's extension, type, display name, and document class.

In some situations, it's worthwhile to subclass `NSDocumentController` in non-`NSDocument`-based apps to get some of its features. For example, the `NSDocumentController` management of the Open Recent menu is useful in apps that don't use subclasses of `NSDocument`.

Topics

Obtaining the Shared Document Controller

`class var shared: NSDocumentController`

Returns the shared `NSDocumentController` instance.

Initializing a New NSDocumentController

`init()`

This method is the designated initializer for `NSDocumentController`.

`init?(coder: NSCoder)`

This method initializes a new NSDocumentController from the coder.

Creating and Opening Documents

`func document(for: URL) -> NSDocument?`

Returns, for a given URL, the open document whose file or file package is located by the URL, or `nil` if there is no such open document.

`func duplicateDocument(withContentsOf: URL, copying: Bool, displayName: String?) throws -> NSDocument`

Creates a new document by reading the contents for the document from another URL, presents its user interface, and returns the document if successful.

`func openDocument(withContentsOf: URL, display: Bool, completionHandler: (NSDocument?, Bool, (any Error)?) -> Void)`

Opens a document located by a URL, optionally presents its user interface, and calls the passed-in completion handler.

`func openUntitledDocumentAndDisplay(Bool) throws -> NSDocument`

Creates a new untitled document, presents its user interface if `displayDocument` is `true`, and returns the document if successful.

`func makeDocument(for: URL?, withContentsOf: URL, ofType: String) throws -> NSDocument`

Instantiates a document located by a URL, of a specified type, but by reading the contents for the document from another URL, and returns it if successful.

`func makeDocument(withContentsOf: URL, ofType: String) throws -> NSDocument`

Instantiates a document located by a URL, of a specified type, and returns it if successful.

```
func makeUntitledDocument(ofType: String) throws -> NSDocument
```

Instantiates a new untitled document of the specified type and returns it if successful.

```
func reopenDocument(for: URL?, withContentsOf: URL, display: Bool,
completionHandler: (NSDocument?, Bool, (any Error)?) -> Void)
```

Reopens a document, optionally located by a URL, by reading the contents for the document from another URL, optionally presents its user interface, and calls the passed-in completion handler.

Managing Documents

```
var documents: [NSDocument]
```

The document objects managed by the receiver.

```
func addDocument(NSDocument)
```

Adds the given document to the list of open documents.

```
var currentDocument: NSDocument?
```

The document object associated with the main window.

```
func document(for: NSWindow) -> NSDocument?
```

Returns the document object whose window controller owns a specified window.

```
var hasEditedDocuments: Bool
```

A Boolean value indicating whether the receiver has any documents with unsaved changes.

```
func removeDocument(NSDocument)
```

Removes the given document from the list of open documents.

Managing Document Types

```
var documentClassNames: [String]
```

An array of strings representing the custom document classes supported by this app.

```
var defaultType: String?
```

Returns the name of the document type that should be used when creating new documents.

```
func documentClass(forType: String) -> AnyClass?
```

Returns the `NSDocument` subclass associated with a given document type.

```
func displayName(forType: String) -> String?
```

Returns the descriptive name for the specified document type, which is used in the File Format pop-up menu of the Save As dialog.

```
func typeForContents(of: URL) throws -> String
```

Returns, for a specified URL, the document type identifier to use when opening the document at that location, if successful.

Autosaving

```
var autosavingDelay: TimeInterval
```

The time interval (in seconds) for periodic autosaving.

Closing Documents

```
func closeAllDocuments(withDelegate: Any?, didCloseAllSelector: Selector?, contextInfo: UnsafeMutableRawPointer?)
```

Iterates through all the open documents and tries to close them one by one using the specified delegate.

```
func reviewUnsavedDocuments(withAlertTitle: String?, cancellable: Bool, delegate: Any?, didReviewAllSelector: Selector?, contextInfo: UnsafeMutableRawPointer?)
```

Displays an alert asking if the user wants to review unsaved documents, quit regardless of unsaved documents, or cancel the save operation.

Responding to Action Messages

```
func newDocument(Any?)
```

An action method called by the New menu command, this method creates a new `NSDocument` object and adds it to the list of such objects managed by the document controller.

```
func openDocument(Any?)
```

An action method called by the Open menu command, it runs the modal Open panel and, based on the selected filenames, creates one or more `NSDocument` objects from the contents of the files.

```
func saveAllDocuments(Any?)
```

As the action method called by the Save All command, saves all open documents of the application that need to be saved.

Managing the Open Dialog

```
func beginOpenPanel(completionHandler: ([URL]?) -> Void)
```

Presents an Open dialog and delivers the results to a completion handler as an array of URLs for the chosen files, or nil.

```
func beginOpenPanel(NSOpenPanel, forTypes: [String]?, completionHandler
: (Int) -> Void)
```

Presents a nonmodal Open dialog that displays files you can open from a list of UTIs.

```
func runModalOpenPanel(NSOpenPanel, forTypes: [String]?) -> Int
```

Presents a modal Open dialog and limits selection to specific file types.

```
var currentDirectory: String?
```

The directory path to use as the starting point in the Open dialog.

```
func urlsFromRunningOpenPanel() -> [URL]?
```

An array of URLs that correspond to the selected files in a running Open dialog.

Managing the Open Recent Menu

```
var maximumRecentDocumentCount: Int
```

The maximum number of items that may be presented in the standard Open Recent menu.

```
func clearRecentDocuments(Any?)
```

Empties the recent documents list for the application.

```
func noteNewRecentDocumentURL(URL)
```

Adds or replaces an Open Recent menu item corresponding to the data located by the URL.

```
func noteNewRecentDocument(NSDocument)
```

Adds or replaces an Open Recent menu item corresponding to the document.

```
var recentDocumentURLs: [URL]
```

The list of recent-document URLs.

Validating User Interface Items

`func validateUserInterfaceItem(any NSValidatedUserInterfaceItem) -> Bool`

Returns a Boolean value that indicates whether a given user interface item should be enabled.

Sharing

`var allowsAutomaticShareMenu: Bool`

A Boolean value that the system uses to insert a Share menu in the File menu.

`func standardShareMenuItem() -> NSMenuItem`

Returns a menu item that your app uses for sharing the current document.

Handling Errors

`func presentError(any Error) -> Bool`

Presents an error alert to the user as a modal panel.

`func presentError(any Error, modalFor: NSWindow, delegate: Any?, didPresent: Selector?, contextInfo: UnsafeMutableRawPointer?)`

Presents an error alert to the user as a modal panel.

`func willPresentError(any Error) -> any Error`

Indicates an error condition and provides the opportunity to return the same or a different error.

Relationships

Inherits From

`NSObject`

Conforms To

`CVarArg`
`Copyable`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`

`Hashable`
`NSCoding`
`NSMenuItemValidation`
`NSObjectProtocol`
`NSUserInterfaceValidations`
`NSWindowRestoration`
`Sendable`

See Also

Documents

{} Developing a Document-Based App

Write an app that creates, manages, edits, and saves text documents.

`class` `NSDocument`

An abstract class that defines the interface for macOS documents.

`class` `NSPersistentDocument`

A document object that can integrate with Core Data.