Class

NSWritingToolsCoordinator.Context

A data object that you use to share your custom view's text with Writing Tools.

macOS 15.2+

```
class Context
```

Mentioned in

Adding Writing Tools support to a custom AppKit view

Overview

At the start of every Writing Tools operation, you create one or more `NSWritingToolsCoordinator.Context` objects with a copy of the text you want Writing Tools to evaluate. Each Writing Tools operation starts with a call to the `writingToolsCoordinator(_:requestsContextsFor:completion:)` method of your `NSWritingToolsCoordinator.Delegate` object. Use the parameters of that method to determine how much of your view's text to provide. For some operations, Writing Tools asks for all of your view's text, but in others it asks for only a portion of the text. When Writing Tools finishes its evaluation, it reports changes back to your delegate relative to the context objects you provided.

When Writing Tools asks for your view's text, create one or more `NSWritingToolsCoordinator.Context` objects with the requested content. If your view contains only one text storage object, create only one context object for the request. However, if you use multiple text storage objects to manage different parts of your view's content, you might need to create multiple context objects. The actual number depends on how much of your text Writing Tools asks for. For example, when Writing Tools asks for all of your view's content, you return one context object for each text storage object in your view. However, if Writing Tools asks for the current

selection, and one text storage object contains all of the selected text, you create only one context object for the content.

Writing Tools uses your context objects as the starting point for its evaluations, and as a reference point for any changes. Because Writing Tools doesn't know anything about your view or its content, it makes suggestions only relative to your context objects. It's your responsibility to take those suggestions and incorporate them back into your view's text storage. In some cases, you might need to store additional information to update your storage correctly. For example, you might need to store, and update as needed, the offset from the start of your document to the start of the text in your context object.

When Writing Tools asks for the currently selected text in your view, include some of the surrounding text in your context object as well. Supply a string that includes the selection and any text up to the nearest paragraph boundary. When creating your context object, specify a range value that represents the portion of that string that corresponds to the text selection. Providing some additional text in your context object can help Writing Tools improve its evaluation of your content. Writing Tools uses the `resolvedRange` property of your context object to indicate what text it considered.

If your context object includes text that you don't want Writing Tools to evaluate, add the `excludeFromWritingTools` attribute to the corresponding characters of your `NSAttributedString` object. You might add this attribute if the text string includes a code listing or readonly content that you don't want Writing Tools to change.

Topics

Creating a context object

`init(attributedString: NSAttributedString, range: NSRange)`

Creates a context object with the specified attributed string and range information.

Getting the source text details

`var attributedString: NSAttributedString`

The portion of your view's text to evaluate.

`var range: NSRange`

The unique identifier of the context object.

Getting the assessed text range

`var resolvedRange: NSRange`

The actual range of text that Writing Tools might change, which can be different than the range of text you supplied.

Identifying the context object

`var identifier: UUID`

The unique identifier of the context object.

Relationships

Inherits From

`NSObject`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSObjectProtocol`
`Sendable`
`SendableMetatype`

See Also

Writing Tools for custom views

- Supporting Writing Tools via the pasteboard

  Adopt a simplified version of the Writing Tools experience in a custom view using the pasteboard and macOS services.

- Adding Writing Tools support to a custom AppKit view

Integrate Writing Tools support, including support for inline replacement animations, to your custom text views on macOS.

`class` NSWritingToolsCoordinator

An object that manages interactions between Writing Tools and your custom text view.

`protocol` Delegate

An interface that you use to manage interactions between Writing Tools and your custom text view.

`class` AnimationParameters

An object you use to configure additional tasks or animations to run alongside the Writing Tools animations.

{} Enhancing your custom text engine with Writing Tools

Add Writing Tools support to your custom text engine to enhance the text editing experience.