Sample Code

Supporting desktop-class features in your iPad app

Enhance your iPad app by adding desktop-class features and document support.

Download

iOS 17.0+ | iPadOS 17.0+ | Mac Catalyst 17.0+ | Xcode 15.0+

Overview

This sample project shows how to build an app that takes full advantage of the iPad's desktop-class features like hardware keyboard and trackpad support, Stage Manager, and rich document-editing capabilities. The app is a markup document editor that allows creating, editing, previewing, and saving documents, including features like a title menu with custom actions, document renaming, Find and Replace, and more.

> **Note**
>
> This sample code project is associated with WWDC22 sessions: 10069: Meet desktop class iPad, 10070: Build a desktop class iPad app, and 10076: Bring your iOS app to the Mac.

Choose the appropriate navigation style

Because the app allows focused viewing and editing of individual documents, it uses the `UINavigationItem.ItemStyle.editor` navigation style. This navigation style moves the navigation item's title to the leading edge and opens up space in the center of the bar for common document actions. To choose the navigation style, the editor view controller assigns the navigation item's `style` property.

Add center items for quick access to common actions

Center item groups are groups of controls that appear in the navigation bar to provide quick access to the app's most important capabilities. A person can customize the navigation bar's center items by moving, removing, or adding certain groups. To enable user customization, the app assigns a string to the navigation item's `customizationIdentifier` property.

```
// Set a `customizationIdentifier` and add center item groups.
navigationItem.customizationIdentifier = "editorViewCustomization"
```

The editor view controller configures center items and assigns them to the navigation item's `centerItemGroups` property in `configureCenterItemGroups()`. The editor view controller creates one fixed group that people can't move or remove from the navigation bar for the Sync Scrolling item using `creatingFixedGroup()`.

```
UIBarButtonItem(primaryAction: UIAction(title: "Sync Scrolling", image: syncScrollir
    syncScrolling.toggle()
    if let barButtonItem = action.sender as? UIBarButtonItem {
        barButtonItem.image = syncScrollingImage
    }
}).creatingFixedGroup(),
```

Other center item groups are optional, which means people can customize their placement in the navigation bar. Optional groups that have `isInDefaultCustomization` set to `false` don't appear in the navigation bar by default. They appear in the customization popover that a person can access by choosing the customization option in the overflow menu.

```
UIBarButtonItem(primaryAction: UIAction(title: "Strikethrough", image: UIImage(syste
    insertTag(.strikethrough)
}).creatingOptionalGroup(customizationIdentifier: "strikethrough", isInDefaultCustom
```

Add a title menu with system and custom actions

A title menu appears when a person taps the navigation item's title. This menu can surface actions that are relevant to the current document. To configure a title menu, the editor view controller assigns a closure to the navigation item's `titleMenuProvider` property.

```
navigationItem.titleMenuProvider = { suggested in
    let custom = [
        UIMenu(title: "Export…", image: UIImage(systemName: "arrow.up.forward.square
```

```
            UIAction(title: "HTML", image: UIImage(systemName: "safari")) { [unowned
                previewView.exportAsWebArchive(named: document.localizedName, preser
            },
            UIAction(title: "PDF", image: UIImage(systemName: "doc.richtext")) { [un
                previewView.exportAsPDF(named: document.localizedName, presenter: se
            }
        ])
    ]
    return UIMenu(children: suggested + custom)
}
```

The closure returns a menu that combines the suggested system actions and custom actions. The first set of actions in the menu are the `suggested` actions that the system passes in to the closure, including Move and Duplicate. The next set of actions are the `custom` actions that the app defines: exporting the document to HTML and PDF.

Configure a document header

A document header displays helpful information about the current document, such as its title, file type, and size. It also provides a place from which to share or drag and drop the document. To display a document header at the top of the title menu, the editor view controller assigns a `UIDocumentProperties` object to the navigation item's `documentProperties` property.

Enable the system Find and Replace experience

The editor view supports editing the content of the document. Because the editor view is a subclass of `UITextView`, enabling the system Find and Replace experience takes one line of code.

```
// Enable Find and Replace in editor text view and register as its
// delegate.
editorTextView.isFindInteractionEnabled = true
```

Setting the `isFindInteractionEnabled` property enables using standard keyboard shortcuts to find text in a document and quickly replace it using the system-provided Find panel.

Enhance multiple-item selection

The outline view is a collection view that serves as a table of contents for the document, allowing for quick navigation or taking actions on the top-level tags in the document. This view supports an

enhanced multiple-selection experience when a person interacts with the app using a keyboard and pointer. The outline view enables lightweight multiple selection of the tags without placing the collection view into editing mode by setting `allowsMultipleSelection`, `allowsFocus`, and `selectionFollowsFocus` to `true`.

```
// Enable multiple selection.
collectionView.allowsMultipleSelection = true

// Enable keyboard focus.
collectionView.allowsFocus = true

// Allow keyboard focus to drive selection.
collectionView.selectionFollowsFocus = true
```

A person can use the keyboard and pointer to select tags, and perform a secondary click to open a context menu with relevant actions. The outline view presents a specialized context menu according to the number of tags in the selection by implementing `collectionView(_:contextMenuConfigurationForItemsAt:point:)` to return different configurations when the selection contains one or many tags.

```
if indexPaths.count > 1 {
    // Action titles for a multiple-item menu.
    hideTitle = "Hide Selected"
    deleteTitle = "Delete Selected"
} else {
    // Action titles for a single-item menu.
    hideTitle = "Hide"
    deleteTitle = "Delete"
}
```

Perform a primary action when tapping a single item

In addition to performing a secondary click on a tag in the outline view, a person can tap a single tag to scroll to its corresponding location in the editor view. To distinguish the explicit user action of tapping one tag to navigate to that location in the document from selecting multiple tags, the outline view implements `collectionView(_:performPrimaryActionForItemAt:)`. The system calls this method when a person taps a single tag without extending a multiple selection of tags.

```
func collectionView(_ collectionView: UICollectionView, performPrimaryActionForItemA
    // Get the element at the `indexPath`.
    if let element = dataSource.itemIdentifier(for: indexPath) {
        delegate?.outline(self, didChoose: element)
    }

    // Wait a short amount of time before deselecting the cell for visual clarity.
    DispatchQueue.main.asyncAfter(deadline: .now() + 0.2) {
        collectionView.deselectItem(at: indexPath, animated: true)
    }
}
```

See Also

iPad, Mac, and Apple Vision Pro

- Building a desktop-class iPad app

  Optimize your iPad app's user experience by adopting desktop-class enhancements for multitasking with Stage Manager, document interactions, text editing, search, and more.

- Elevating your iPad app with a tab bar and sidebar

  Provide a compact, ergonomic tab bar for quick access to key parts of your app, and a sidebar for in-depth navigation.

- Multitasking on iPad, Mac, and Apple Vision Pro

  Implement multitasking APIs to seamlessly integrate your app with iPadOS, macOS, and visionOS.