Sample Code

Detecting changes in the preferences window

Listen for and respond to a user's preference changes in your Mac app built with Mac Catalyst using Combine.

Download

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | Xcode 11.1+

Overview

With Combine, your app can listen for changes a user makes to the app's Preferences window, and respond to those changes. The sample app provides a Preferences window with one setting: background color. When the user selects a color, the background of the main view changes to match their selection.

This sample code project shows how to:

- Add a Preferences window in a Mac app built with Mac Catalyst.
- Register default values for the preferences.
- Retrieve current preference values.
- Listen for and respond to changes the user makes in the Preferences window.

To use the sample app, open the sample code project in Xcode and select My Mac as the destination. Then, build and run the sample project.

Provide a preferences window in the app

The sample app includes a `Settings.bundle` file that the system uses to automatically add the standard Preferences menu item to the app menu. Selecting the menu item displays a Preferences window that the system generates based on the preference specifiers defined in the Settings bundle. To learn more, see Displaying a Preferences window.

The Settings bundle for the sample app has a preference specifier for setting the background color of the main view. It also has a child pane preference specifier, which displays a second tab of preferences in the Preferences windows. The Settings bundle file `Root.plist` defines these specifiers, while the file `OtherSettings.plist` defines the preference specifiers for the child pane.

Register default preference values

When the user changes preferences in the Preferences window, the window saves them to the application domain of the user defaults system. To store and retrieve the preference values within the app, the sample app uses `UserDefaults`. However, when you launch the sample app for the first time, the preference values don't exist in the user defaults system. If the app tries retrieving a value, `UserDefaults` returns `nil`.

To ensure that the app always retrieves a non-`nil` value, the sample app registers the default preference values with the registration domain. However, this domain doesn't persist these values between app launches, so the sample app registers the default values each time the user launches the app.

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchO

    // To ensure that the app has a good set of preference values, register
    // the default values each time the app launches.
    registerDefaultPreferenceValues()

    return true
}
```

The method `registerDefaultPreferenceValues()` retrieves the default values from the Settings bundle by retrieving the preference specifiers from the `Root.plist` file and parsing the specifiers for their default value. After retrieving the values, the method registers the default values.

```
func registerDefaultPreferenceValues() {
    let preferenceSpecifiers = retrieveSettingsBundlePreferenceSpecifiers(from: "Roo
    let defaultValuesToRegister = parse(preferenceSpecifiers)
```

```
    // Register the default values with the registration domain.
    UserDefaults.standard.register(defaults: defaultValuesToRegister)
}
```

To parse the preference specifiers, the `parse()` method loops through the array of specifiers, copying the default values into the dictionary `defaultValuesToRegister`. If the method detects the `PSChildPaneSpecifier` type, it gets the name of the child pane property list file, and merges the default values in the file into the `defaultValuesToRegister` dictionary. After gathering the default values, the method returns the dictionary to the caller.

```
func parse(_ preferenceSpecifiers: [NSDictionary]) -> [String: Any] {
    var defaultValuesToRegister = [String: Any]()

    // Parse the preference specifiers, copying the default values
    // into the `defaultValuesToRegister` dictionary.
    for preferenceItem in preferenceSpecifiers {
        if let key = preferenceItem["Key"] as? String,
            let defaultValue = preferenceItem["DefaultValue"] {
            defaultValuesToRegister[key] = defaultValue
        }

        // Add child pane preference specifiers.
        if let type = preferenceItem["Type"] as? String,
            type == "PSChildPaneSpecifier" {
            if var file = preferenceItem["File"] as? String {
                if file.hasSuffix(".plist") == false {
                    file += ".plist"
                }
                let morePreferenceSpecifiers = retrieveSettingsBundlePreferenceSpeci
                let moreDefaultValuesToRegister = parse(morePreferenceSpecifiers)
                defaultValuesToRegister.merge(moreDefaultValuesToRegister) { (currer
            }
        }
    }

    return defaultValuesToRegister
}
```

Retrieve preference values

After registering the default values with the registration domain, the app can retrieve a preference value without the possibility of encountering an unavailable value. To simplify access to the

background color preference value, the sample app extends `UserDefaults` to include properties for each preference value.

```
extension UserDefaults {

    @objc dynamic var backgroundColorValue: Int {
        return integer(forKey: "backgroundColorValue")
    }

    @objc dynamic var someRandomOption: Bool {
        return bool(forKey: "someRandomOption")
    }

}
```

Handle changes made in the preferences window

As the user changes the background color setting in the Preferences window, the app changes the background color of its main view. To accomplish this, the view controller for the main view creates a subscriber in the `viewDidLoad()` method. When the background color value changes, the subscriber receives the new value, maps it to a `UIColor` object, and assigns the color to the view's `backgroundColor` property.

```
var subscriber: AnyCancellable?   // Subscriber of preference changes.

override func viewDidLoad() {
    super.viewDidLoad()

    // Set the view's initial background color to the color specified in Preferences
    if let colorSetting = BackgroundColors(rawValue: UserDefaults.standard.backgrour
        view.backgroundColor = colorSetting.currentColor()
    }

    // Listen for changes to the background color preference made in the Preferences
    subscriber = UserDefaults.standard
        .publisher(for: \.backgroundColorValue, options: [.initial, .new])
        .map( { BackgroundColors(rawValue: $0)?.currentColor() })
        .assign(to: \UIView.backgroundColor, on: self.view)
}
```

See Also

User preferences

Displaying a Preferences window

Provide a Preferences window in your Mac app built with Mac Catalyst so users can manage app preferences defined in a Settings bundle.