Sample Code

Add Home Screen quick actions

Expose commonly used functionality with static or dynamic 3D Touch Home Screen quick actions.

Download

iOS 13.0+ | iPadOS 13.0+ | Xcode 12.0+

Overview

On the Home screen of a device running iOS 13 or later, apps can display Home Screen quick actions when users touch and hold the app icon (on a 3D Touch device, users press briefly on the icon).

Each Home Screen quick action includes a title, an icon on the left or right (depending on your app's position on the Home Screen), and an optional subtitle. Define quick actions statically at build time from the app's `Info.plist` or dynamically configured at runtime. For information about the types of functionality you might expose using quick actions, see the Human Interface Guidelines.

The app in this sample project is a basic contact manager that allows users to view and edit a small set of contacts. The initial view controller shows a list of contacts in a table view. Tapping any contact shows a detail screen where you can edit the name and email address, or tap to turn on the Favorite switch.

When you press the sample app's Home Screen icon, several Home Screen quick actions are displayed. The first two give access to search and sharing functionality, and the rest provide shortcuts to favorite contacts.

Define static quick actions

If the quick actions appropriate for an app never change, define them as static quick actions using the project's `Info.plist` file. This sample project defines two static quick actions: one for searching and one for sharing. The `UIApplicationShortcutItems` key in the `Info.plist` file contains an array of two dictionaries that represent the two static quick actions.

```
<key>UIApplicationShortcutItems</key>
<array>
    <dict>
        <key>UIApplicationShortcutItemType</key>
        <string>SearchAction</string>
        <key>UIApplicationShortcutItemIconType</key>
        <string>UIApplicationShortcutIconTypeSearch</string>
        <key>UIApplicationShortcutItemTitle</key>
        <string>Search</string>
        <key>UIApplicationShortcutItemSubtitle</key>
        <string>Search for an item</string>
    </dict>
    <dict>
        <key>UIApplicationShortcutItemType</key>
        <string>ShareAction</string>
        <key>UIApplicationShortcutItemIconType</key>
        <string>UIApplicationShortcutIconTypeShare</string>
        <key>UIApplicationShortcutItemTitle</key>
        <string>Share</string>
        <key>UIApplicationShortcutItemSubtitle</key>
        <string>Share an item</string>
    </dict>
</array>
```

The value for `UIApplicationShortcutItemType` must be a unique string that the system passes to your app when a user invokes the quick action. When processing the selected quick action, the unique item type distinguishes between the defined quick actions.

For information about other `Info.plist` keys available for configuring Home Screen quick actions, see the Information Property List Key Reference.

Define dynamic quick actions

Dynamic quick actions depend on specific data or state of the app. Configure dynamic Home Screen quick actions by setting the `shortcutItems` property of the shared `UIApplication` instance to an array of `UIApplicationShortcutItem` objects.

Set dynamic screen quick actions at any point, but the sample sets them in the `sceneWillResignActive(_:)` function of the scene delegate. During the transition to a background state is a good time to update any dynamic quick actions, because the system executes this code before the user returns to the Home Screen.

```
func sceneWillResignActive(_ scene: UIScene) {
    // Transform each favorite contact into a UIApplicationShortcutItem.
    let application = UIApplication.shared
    application.shortcutItems = ContactsData.shared.favoriteContacts.map { contact -
        return UIApplicationShortcutItem(type: ActionType.favoriteAction.rawValue,
                                         localizedTitle: contact.name,
                                         localizedSubtitle: contact.email,
                                         icon: UIApplicationShortcutIcon(systemImage
                                         userInfo: contact.quickActionUserInfo)
    }
}
```

Don't limit the number of quick actions provided to the `shortcutItems` property, because the system displays only the number of items that fit the screen. For this reason, the implementation of `favoriteContacts` on the `ContactsData` class returns all contacts that have the `favorite` property set to `true`.

Define quick action icons

Define quick action icons with `UIApplicationShortcutIcon` class defined two different ways: with a template image or SF Symbol.

With a template image name:

```
UIApplicationShortcutIcon(type: .favorite)
```

With an SF Symbol name:

```
UIApplicationShortcutIcon(systemImageName: "star.fill")
```

As mentioned earlier, this sample app creates two static quick actions in its `Info.plist` with their icons set to `UIApplicationShortcutItemIconType`.

```
<key>UIApplicationShortcutItemIconType</key>
<string>UIApplicationShortcutIconTypeSearch</string>
```

Create a quick action icon with an SF Symbol:

```
<key>UIApplicationShortcutItemIconSymbolName</key>
<string>square.stack.3d.up</string>
```

To support older versions of iOS, you can specify multiple icon-related keys in the app's `Info.plist` for a given `UIApplicationShortcutItem`. iOS chooses the icon in the following order:

1. UIApplicationShortcutItemIconSymbolName
2. UIApplicationShortcutItemIconFile
3. UIApplicationShortcutItemIconType

iOS prefers the symbol name icon if it's defined, otherwise, the icon file if defined, otherwise the last choice is the icon type.

Pass data with a quick action

The dynamic quick actions in this sample all have the same `type` because they all perform the same action. However, the contact information associated with each is different and the sample passes it to through the `userInfo` dictionary.

```
var quickActionUserInfo: [String: NSSecureCoding] {
    /** Encode the name of the contact into the userInfo dictionary so it can be pas
        back when a quick action is triggered. Note: In the real world, it's more ap
        to encode a unique identifier for the contact than for the name.
    */
    return [ SceneDelegate.favoriteIdentifierInfoKey: self.identifier as NSSecureCod
}
```

Static Home Screen quick actions can also pass `userInfo` data by including it in the `UIApplicationShortcutItemUserInfo` key in the app's `Info.plist` file.

All values passed as `userInfo` must conform to `NSSecureCoding`.

Respond to triggered quick actions

When the user triggers a Home Screen quick action, the app is notified in one of these ways:

- If the app isn't already loaded, it's launched and passes details of the shortcut item in through the connectionOptions parameter of the `scene(_:willConnectTo:options:)` function.

```
func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connecti
    /** Process the quick action if the user selected one to launch the app.
        Grab a reference to the shortcutItem to use in the scene.
    */
    if let shortcutItem = connectionOptions.shortcutItem {
        // Save it off for later when we become active.
        savedShortCutItem = shortcutItem
    }
}
```

- If your app is already loaded, the system calls the `windowScene(_:performActionFor:completionHandler:)` function of your scene delegate.

```
func windowScene(_ windowScene: UIWindowScene,
                 performActionFor shortcutItem: UIApplicationShortcutItem,
                 completionHandler: @escaping (Bool) -> Void) {
    let handled = handleShortCutItem(shortcutItem: shortcutItem)
    completionHandler(handled)
}
```

When this sample handles the quick action, it shows an alert, but in a real-world app the actual functionality would be executed.

Debug quick actions on app launch

For Xcode to debug quick actions on app launch, an app needs to have the target's scheme configured. Follow these instructions to debug quick actions at launch:

1. Build and run the app to make sure it's installed on the device.
2. Quit the app.
3. Set a breakpoint in SceneDelegate.swift, function `scene(_:willConnectTo:options:)` to be used when a shortcut will be chosen by the user.
4. In Xcode find the target's Scheme: Open Product -> Scheme -> Edit Scheme
5. Select the Run Scheme, Info tab.
6. Change the Launch setting to "Wait for executable to be launched".
7. Run the app again in Xcode (Xcode's debugger will then wait).
8. Go to the Home Screen, tap and hold on the sample's app icon.

9. Select a quick action shortcut.

The breakpoint will then fire.

See Also

Home Screen quick actions

`class` **`UIApplicationShortcutItem`**

An application shortcut item, also called a Home Screen dynamic quick action, that specifies a user-initiated action for your app.

`class` **`UIApplicationShortcutIcon`**

An image you can optionally associate with a Home Screen quick action to improve its appearance and usability.

`class` **`UIMutableApplicationShortcutItem`**

A mutable Home Screen dynamic quick action, which is an item that specifies a configurable user-initiated action for your app.