Sample Code

Fruta: Building a feature-rich app with SwiftUI

Create a shared codebase to build a multiplatform app that offers widgets and an App Clip.

Download

iOS 15.4+ | iPadOS 15.4+ | macOS 12.3+ | Xcode 13.3+

Overview

Note

This sample project is associated with WWDC21 sessions 10107: Platforms State of the Union, 10012: What's New in App Clips, 10013: Build Light and Fast App Clips, 10220: Localize your SwiftUI App.

It's also associated with WWDC20 sessions 10637: Platforms State of the Union, 10146: Configure and Link Your App Clips, 10120: Streamline Your App Clip, 10118: Create App Clips for Other Businesses, 10096: Explore Packages and Projects with Xcode Playgrounds, and 10028: Meet WidgetKit.

The Fruta sample project builds an app for macOS, iOS, and iPadOS that implements SwiftUI platform features like widgets, App Clips, and localization. Users can order smoothies, save favorite drinks, collect rewards, and browse recipes. It contains two flavors of app targets:

- Simple iOS and macOS app targets that you build using Personal Team signing. This iOS app runs in Simulator, and only requires a standard Apple ID to install on a device. The simple app implements a rich, localized SwiftUI interface. Users can browse and order smoothies, and save favorite drinks.

- Full featured iOS All and macOS All app targets. The full iOS app runs in Simulator, and on devices with an Apple Developer membership. This app includes widget extensions that enable users to add a widget to their iOS Home Screen or the macOS Notification Center, and to view their rewards or a favorite smoothie. This app also embeds an App Clip. With the App Clip, users can discover and instantly launch some of the app's functionality on their iPhone or iPad without installing the app.

The Fruta sample app leverages Sign in with Apple and PassKit (Apple Pay and Wallet) to provide a streamlined user experience.

Configure the sample code project

To build this project for iOS 15.4, use Xcode 13.3. The runtime requirement is iOS 15.4. To build this project for macOS 12.3, use Xcode 13.3.

To configure the iOS and macOS app targets without an Apple Developer account, follow these steps:

1. In the targets' Signing & Capabilities panes click Add Account, and log in with your Apple ID.
2. Chose the Your Name (Personal Team) from the team drop down menu.
3. Click build-and-run.
4. On iOS and iPadOS devices you need to navigate to Settings > General > VPN & Device Management and trust your developer certificate.

To configure the iOS All and macOS All apps, follow these steps:

1. To run on your devices, including on macOS, set your team in the targets' Signing & Capabilities panes. Xcode manages the provisioning profiles for you.
2. To run on an iOS or iPadOS device, open the `iOSClip.entitlements` file and update the value of the `Parent Application Identifiers Entitlement` to match the iOS app's bundle identifier.
3. Make a note of the App Group name on the iOS target's Signing & Capabilities tab in Project Settings. Substitute this value for group.example.fruta in the `Model.swift` file.
4. To enable the in-app-purchase flow, edit the Fruta iOS "Run" scheme, and select `Configuration.storekit` for StoreKit Configuration.

Create a shared codebase in SwiftUI

To create a single app definition that works for multiple platforms, the project defines a structure that conforms to the App protocol. Because the `@main` attribute precedes the structure definition, the system recognizes the structure as the entry point into the app. Its computed body property

returns a `WindowGroup` scene that contains the view hierarchy displayed by the app to the user. SwiftUI manages the presentation of the scene and its contents in a platform-appropriate manner.

```
@main
struct FrutaApp: App {
    @StateObject private var model = Model()

    var body: some Scene {
        WindowGroup {
            ContentView()
                .environmentObject(model)
        }
        .commands {
            SidebarCommands()
            SmoothieCommands(model: model)
        }
    }
}
```

For more information, see App organization.

Offer an App Clip

On iOS and iPadOS, the Fruta app offers some of its functionality to users who don't have the full app installed as an App Clip. The app's Xcode project contains an App Clip target, and, instead of duplicating code, reuses code that's shared across all platforms to build the App Clip. In shared code, the project makes use of the Active Compilation Condition build setting to exclude code for targets that don't define the `APPCLIP` value. For example, only the App Clip target presents an App Store overlay to prompt the user to get the full app.

```
VStack(spacing: 0) {
    Spacer()

    orderStatusCard

    Spacer()

    #if EXTENDED_ALL
    if presentingBottomBanner {
        bottomBanner
```

```
    }
    #endif

    #if APPCLIP
    Text(verbatim: "App Store Overlay")
        .hidden()
        .appStoreOverlay(isPresented: $presentingAppStoreOverlay) {
            SKOverlay.AppClipConfiguration(position: .bottom)
        }
    #endif
}
.onChange(of: model.hasAccount) { _ in
    #if APPCLIP
    if model.hasAccount {
        presentingAppStoreOverlay = true
    }
    #endif
}
```

For more information, see Creating an App Clip with Xcode and Choosing the right functionality for your App Clip.

Create a widget

To allow users to see some of the app's content as a widget on their iOS Home screen or in the macOS Notification Center, the Xcode project contains targets for widget extensions. Both use code that's shared across all targets.

For more information, see WidgetKit.

See Also

Creation

Creating an App Clip with Xcode

Add an App Clip target to your Xcode project and share code between the App Clip and its corresponding full app.

`Parent Application Identifiers Entitlement`

A list of parent application identifiers for an App Clip with exactly one entry.

`com.apple.developer.associated-appclip-app-identifiers`

A list of App Clip identifiers for an app with exactly one entry.

`com.apple.developer.on-demand-install-capable`

A Boolean value that indicates whether a bundle represents an App Clip.