Sample Code

Soup Chef with App Intents: Migrating custom intents

Integrating App Intents to provide your app's actions to Siri and Shortcuts.

Download

iOS 14.3+ | iPadOS 14.3+ | Mac Catalyst 14.3+ | Xcode 17.0+

Overview

This version of Soup Chef highlights the changes required to migrate from a custom intent to App Intents. It modifies the existing Soup Chef sample to add support for AppIntents, AppEntities, and App Shortcuts as described in Migrate Custom Intents to App Intents.

Configure the sample code project

Before you can run Soup Chef, you need to:

1. Set the app group name for the SoupChef, SoupChefIntents, SoupChefWatchExtension, and SoupChefIntentsWatch targets to a valid name. For more information on App Groups, see Configure App Groups.
2. Change the value of AppGroup in UserDefaults+DataSource.swift to match your app group name.

Share code between the app and app extension

The Soup Chef project contains targets for an app and an Intents app extension, which the system uses to handle shortcuts that run in the background. To avoid duplicating code that is common across the two targets, the project includes a shared framework called SoupKit. This framework

provides a central location for shared code responsible for tasks such as data management and donating shortcuts. For more information about structuring code, see Structuring Your Code to Support App Extensions.

Convert the intent definition

The intent definitions, which define the custom intents, are recreated as their respective AppIntent representation using the “Convert to AppIntents” button in the intent definition file. The new intent files, located in the App Intent Group of the project, each have an autogenerated structure and implement the CustomIntentMigratedAppIntent protocol. Each parameter of the custom intent represents an `AppEntity` structure.

```
struct OrderSoup: AppIntent, CustomIntentMigratedAppIntent {
```

Implement queries

Each of the four parameters for the `OrderSoupIntent` have their respective AppEntity representation. The `SoupAppEntity` and `ToppingAppEntity` use the existing data manager to resolve the entity values. Additionally, apps can also resolve entities by string or UUID. The full list of entities the person sees when they tap the parameter in the Shortcuts app can also show by returning a collection of entities in the `suggestedEntities()` implementation. `OrderType` has a representation as `AppEnum`, and `OrderDetails` is represented as a `TransientAppEntity` because it doesn’t have unique identifier.

```
func suggestedEntities() async throws -> IntentItemCollection<SoupAppEntity> {
            let soupMenuManager = SoupMenuManager()

            // Only adopt `EntityStringQuery` for searching large catalogs, not for
            // The Shortcuts app supports filtering of small collections by default.
            let availableRegularItems = soupMenuManager.findItems(exactlyMatching: [
            let availableDailySpecialItems = soupMenuManager.findItems(exactlyMatchi

            return ItemCollection {
                ItemSection<SoupAppEntity>(
                    items:
                        availableRegularItems .map {
                            IntentItem<SoupAppEntity>(
                                SoupAppEntity($0),
                                title: SoupAppEntity($0).localizedStringResource,
                                image: SoupAppEntity($0).displayRepresentation.image
```

```
                    }
                )
                ItemSection<SoupAppEntity>(
                    title: "Specials",
                    items: availableDailySpecialItems .map {
                        IntentItem<SoupAppEntity>(
                            SoupAppEntity($0),
                            title: SoupAppEntity($0).localizedStringResource,
                            image: SoupAppEntity($0).displayRepresentation.image)
                    }
                )
            }
```

Replace custom intents with App Intents

The `Perform` method in AppIntents subsumes the resolve, confirm, and handle functionality that was required with custom Intents. Entities declared in an App Intent can be optional or required. The entity type and query implementation help to resolve their values automatically in App Intents. Handling cases where user input requires confirmation or needs new value are handled explicitly within the `Perform` function. For example, confirming the order before placing it, or requesting a delivery location aren't provided. The `Perform` method handles the final execution and returns an intent result which can be the dialog that needs to be spoken as well as the snippet that is shown onscreen to the person.

The following is an example of how to resolve `Topping` using an Entity Query:

```
func entities(matching string: String) async throws -> [ToppingAppEntity] {
            let menuItemTopping = Order.MenuItemTopping.allCases.filter { $0.rawValu
            let result = menuItemTopping.map { (topping) -> ToppingAppEntity in
                return ToppingAppEntity(
                    id: topping.rawValue, displayString: topping.localizedName(useDe
            }
            return result
        }
```

The following is an example of how to validate quantity values and returning results in perform method:

```
    func perform() async throws -> some ProvidesDialog & ShowsSnippetView {
        if quantity > menuItem[0].itemsInStock {
                let errorPrompt = OrderSoupError.notEnoughInStock(quantity ).localiz
```

```
                return .result(value: OrderDetailsAppEntity(), dialog: I
        }
```

Provide dynamic options

The `storeLocations` parameter has a backing by a `DynamicOptionsProvider` that returns all store locations directly in results method.

```
@available(iOS 16.0, macOS 13.0, watchOS 9.0, tvOS 16.0, *)
struct SoupChefAppShortcutsProvider: AppShortcutsProvider {
    static var appShortcuts: [AppShortcut] {
        AppShortcut(
            intent: OrderSoup(),
            phrases: [
                "Order \(.applicationName)",
                "Order \(\.$soup) from \(.applicationName)"
            ],
            shortTitle: "Order Soup"
        )
    }
    static var shortcutTileColor: ShortcutTileColor = .orange
}
```

App Shortcut provider and localization

To make the Intent easier to discover, this version of Soup Chef implement AppShortcuts that allow the Intent to automatically appear in the Shortcut app. To aid with localization of AppShortcuts, use `AppShortcuts.strings` to add support for other locales.

```
"TOMATO_SOUP" = "Tomato Soup";
"NE_CLAM_CHOWDER" = "New England Clam Chowder";
"CHICKEN_NOODLE_SOUP" = "Chicken Noodle Soup";
```

See Also

Sample code

{} Adding Shortcuts for Wind Down

Reveal your app's shortcuts inside the Health app.

{} Booking Rides with SiriKit

Add Intents extensions to your app to handle requests to book rides using Siri and Maps.

{} Handling Payment Requests with SiriKit

Add an Intent Extension to your app to handle money transfer requests with Siri.

{} Handling Workout Requests with SiriKit

Add an Intent Extension to your app that handles requests to control workouts with Siri.

{} Integrating Your App with Siri Event Suggestions

Donate reservations and provide quick access to event details throughout the system.

{} Managing Audio with SiriKit

Control audio playback and handle requests to add media using SiriKit Media Intents.

{} Providing Hands-Free App Control with Intents

Resolve, confirm, and handle intents without an extension.

{} Soup Chef: Accelerating App Interactions with Shortcuts

Make it easy for people to use Siri with your app by providing shortcuts to your app's actions.