Article

Structuring Your Code to Support App Extensions

Move your back-end services to a private framework so your app and app extensions can use them.

Overview

An app extension is an agent that acts on behalf of your app, vending services or providing information to the system when asked. Because they extend your app’s behavior, app extensions often need access to the same back-end services and data that your app uses.

Lightweight extensions that don’t include most of your app’s code can respond to a user’s request more quickly. Long-running tasks like playing music or tracking a workout need to happen in your app. If necessary, you can resolve the parameters of an intent in an extension, then instruct SiriKit to launch your app for the final handling step.

If your app and app extension share services, consider structuring your code in the following way:

- **Implement your core services in a private shared framework.** A private shared framework lets you place the code for accessing your services in one code module and use that code from multiple targets. Shared frameworks minimize the size of both executables and make testing easier by ensuring that each executable uses the same code path.
- **Use a shared container to store common resources.** Put relevant images and data files into a shared container so your app and app extension can use them. You enable shared container support in the Capabilities tab of each target.

See Also

Articles

Adding User Interactivity with Siri Shortcuts and the Shortcuts App

Add custom intents and parameters to help users interact more quickly and effectively with Siri and the Shortcuts app.

Defining Relevant Shortcuts for the Siri Watch Face

Inform Siri when your app’s shortcuts may be useful to the user.

Deleting Donated Shortcuts

Remove your donations from Siri.

Dispatching intents to handlers

Provide SiriKit with an intent handler capable of handling a specific intent.

Improving Siri Media Interactions and App Selection

Fine-tune voice controls and improve Siri Suggestions by sharing app capabilities, customized names, and listening habits with the system.

Improving interactions between Siri and your messaging app

Donate app-specific content, use Siri’s contact suggestions, and adopt the latest platform features to create a more consistent messaging experience.

Registering Custom Vocabulary with SiriKit

Register your app’s custom terminology, and provide sample phrases for how to use your app with Siri.

Confirming the Details of an Intent

Perform final validation of the intent parameters and verify that your services are ready to fulfill the intent.

Handling an Intent

Fulfill the intent and provide feedback to SiriKit about what you did.

Resolving the Parameters of an Intent

Validate the parameters of an intent and make sure that you have the information you need to continue.

Generating a List of Ride Options

Generate ride options for Maps to display to the user.

Handling the Ride-Booking Intents

Support the different intent-handling sequences for booking rides with Shortcuts or Maps.

Donating Reservations

Inform Siri of reservations made from your app.

Specifying Synonyms for Your App Name

Provide alternative names for your app that are more familiar or easier for users to speak.

Intent Phrases

The keys that you include in your global vocabulary file to show how users engage your app from Siri.