Protocol

SnippetIntent

An app intent that presents an interactive snippet onscreen.

iOS 26.0+ | iPadOS 26.0+ | Mac Catalyst | macOS 26.0+ | tvOS 26.0+ | visionOS 26.0+ | watchOS 26.0+

```
protocol SnippetIntent : AppIntent where Self.PerformResult : ShowsSnippetView
```

Mentioned in

Displaying static and interactive snippets

Overview

An app intent can present custom SwiftUI views to show people the result of their action, confirm a selection, and more. For example, an app could show a confirmation for a successful order.

> **Note**
>
> The system can call a `SnippetIntent` multiple times. For more information, refer to Displaying static and interactive snippets.

By conforming your app intent to the `SnippetIntent` protocol, you can provide a *snippet*, a custom view with interactivity. Similar to widgets and Live Activities, a snippet can include buttons or toggles that use an `AppIntent` for their functionality. In many cases, you might be able to reuse views of your interactive widget or Live Activity.

The following code snippet shows what the perform method for a task management app could look like. The intent asynchronously loads a list of tasks and presents it using the `TodoListView`

that the `perform()` function returns. The `TodoListView` could then offer a toggle for each item in the list to immediately mark a task as completed.

```
struct ShowTodoListIntent: SnippetIntent {
    // ...

    @Parameter var todos: [Todo]

    // ...

    func perform() async throws -> some IntentResult & ShowsSnippetView {
        // Fetch persisted todos to have the right
        // up-to-date state performing this method. Make sure to
        // consider that the system calls perform() several times.
        let currentTodos = await TodoStore().find(todos)
        let snippet = TodoListView(todos: currentTodos)
        return .result(view: snippet)
    }
}
```

When someone interacts with a snippet's button or a toggle, the system first performs its associated app intent. When the button or toggle's intent completes, the system calls the snippet intent's `perform()` method again. In your snippet intent's `perform()` implementation, make sure to handle multiple calls of `perform()`. For example, the example above would need to fetch the list of tasks to make sure it displays the most recent data. If a user completes a task from a snippet, the snippet needs to reflect this change and show the task as completed or remove it from the list of tasks.

Important

Only app intents that conform to this protocol can present views with interactive elements, like buttons and toggles, that work. Additionally, make sure to conform your intent to this protocol so that the system knows to call again your `perform` function to render the new state of the snippet after it performed the action of a button or toggle.

If your intent does more than just returning a snippet; for example, if you extend an app intent that returns a value to also return a snippet; the intent is automatically discoverable in the Shortcuts app and Spotlight. If an app intent conforms to `SnippetIntent` and only returns a snippet — their return type only conforms to `IntentResult` and `ShowsSnippetView` —, it's nondiscoverable by the Shortcuts app and in Spotlight. To make such an intent discoverable, explicitly set `isDiscoverable` to `true`.

Topics

Default implementation

`struct EmptySnippetIntent`

A snippet intent that renders an empty view.

Type Methods

`static func reload()`

Refreshes the intent's snippet presentation.

Relationships

Inherits From

`AppIntent`
`PersistentlyIdentifiable`
`Sendable`
`SendableMetatype`

Conforming Types

`EmptySnippetIntent`

See Also

Interactive Snippets

Displaying static and interactive snippets

Enable people to view the outcome of an app intent and immediately perform follow-up actions.