Framework

Observation

Make responsive apps that update the presentation when underlying data changes.

iOS 17.0+ | iPadOS 17.0+ | Mac Catalyst 17.0+ | macOS 14.0+ | tvOS 17.0+ | visionOS 1.0+ | watchOS 10.0+

Overview

Observation provides a robust, type-safe, and performant implementation of the observer design pattern in Swift. This pattern allows an observable object to maintain a list of observers and notify them of specific or general state changes. This has the advantages of not directly coupling objects together and allowing implicit distribution of updates across potential multiple observers.

The Observation frameworks provides the following capabilities:

- Marking a type as observable
- Tracking changes within an instance of an observable type
- Observing and utilizing those changes elsewhere, such as in an app's user interface

To declare a type as observable, attach the `Observable()` macro to the type declaration. This macro declares and implements conformance to the `Observable` protocol to the type at compile time.

```
@Observable
class Car {
    var name: String = ""
    var needsRepairs: Bool = false

    init(name: String, needsRepairs: Bool = false) {
        self.name = name
        self.needsRepairs = needsRepairs
    }
}
```

To track changes, use the `withObservationTracking(_:onChange:)` function. For example, in the following code, the function calls the `onChange` closure when a car's name changes. However, it doesn't call the closure when a car's `needsRepair` flag changes. That's because the function only tracks properties read in its `apply` closure, and the closure doesn't read the `needsRepair` property.

```
func render() {
    withObservationTracking {
        for car in cars {
            print(car.name)
        }
    } onChange: {
        print("Schedule renderer.")
    }
}
```

Topics

Observable conformance

`macro Observable()`

Defines and implements conformance of the Observable protocol.

`protocol Observable`

A type that emits notifications to observers when underlying data changes.

Change tracking

`func withObservationTracking<T>(() -> T, onChange: @autoclosure () -> () -> Void) -> T`

Tracks access to properties.

`struct ObservationRegistrar`

Provides storage for tracking and access to data changes.

Observation in SwiftUI

{} Managing model data in your app

Create connections between your app's data model and views.

{} Migrating from the Observable Object protocol to the Observable macro

Update your existing app to leverage the benefits of Observation in Swift.

Structures

`struct Observations`

An asychronous sequence generated from a closure that tracks the transactional changes of `@Observable` types.

Macros

`macro ObservationIgnored()`

Disables observation tracking of a property.

`macro ObservationTracked()`

Synthesizes a property for accessors.