Article

Reverting data changes using the undo manager

Automatically record data change operations that people perform in your SwiftUI app, and let them undo and redo those changes.

Overview

People who interact with apps to change data typically expect the app to provide a way to revert changes that they no longer wish to make. To provide this *undo* support, apps can use `UndoManager`, which can:

- Record operations that a person performs while using the app
- Provide undo capability to revert the last operation
- Provide redo capability to reapply the last reverted operation

If your app uses SwiftData, it can automatically register data changes with the undo manager. No need for you to add more code to your app that records those changes. Instead, just enable undo in SwiftData.

Enable undo in SwiftData

Using SwiftData to store data, you can enable undo support for data changes by setting the `isUndoEnabled` parameter to `true` when applying the `modelContainer(for:inMemory:isAutosaveEnabled:isUndoEnabled:onSetup:)` modifier to a scene or view in your SwiftUI app:

```
@main
struct SwiftDataAnimalsApp: App {
    var body: some Scene {
```

```
        WindowGroup() {
            ContentView()
        }
        .modelContainer(for: AnimalCategory.self, isUndoEnabled: true)
    }
}
```

By default, the `isUndoEnabled` parameter value is `false`, which disables undo support in SwiftData. If you set this parameter to `true` in your app, the app's window or window group provides the system's undo manager to SwiftData. SwiftData then binds the undo manager to the `mainContext` of the `ModelContainer`. When someone makes data changes in your app and the app saves those changes to the main context, they can use system gestures in iOS and iPadOS like three finger swap and device shake, or choose Edit > Undo or Edit > Redo in macOS, to undo and redo those changes, respectively.

> **Note**
>
> To retrieve the `mainContext` in a SwiftUI view, use the `modelContext` environment value; for example, `@Environment(\.modelContext) private var modelContext`.

Other than enabling undo in SwiftData, you don't need to write any additional code to manage undo operations for data changes. Instead, SwiftData automatically registers data change operations with the undo manager each time SwiftData saves those changes to the main context. And if your app uses `DocumentGroup` to manage data storage, undo and redo in SwiftData is automatically enabled.

> **Important**
>
> Setting `isUndoEnabled` to `true` provides automatic undo and redo support in your app for data changes it saves to the main context only. This setting doesn't provide undo and redo support for changes saved to other model contexts that your app may use, such as a background model context that stores data retrieved from an external source.

See Also

Model life cycle

`class` `ModelContainer`

An object that manages an app's schema and model storage configuration.

`class` **`ModelContext`**

An object that enables you to fetch, insert, and delete models, and save any changes to disk.

Fetching and filtering time-based model changes

Track all inserts, updates, and deletes that occur in a data store and process them as a series of chronological transactions.

`struct` **`HistoryDescriptor`**

A type that describes the criteria, and, optionally, sort order, to use when fetching history data

{} Deleting persistent data from your app

Explore different ways to use SwiftData to delete persistent data.

Syncing model data across a person's devices

Add the required capabilities and define a compatible schema to enable SwiftData to automatically sync your app's model data using iCloud.

Concurrency support

Types you use to access model attributes and perform storage-related tasks in a safe and isolated way.