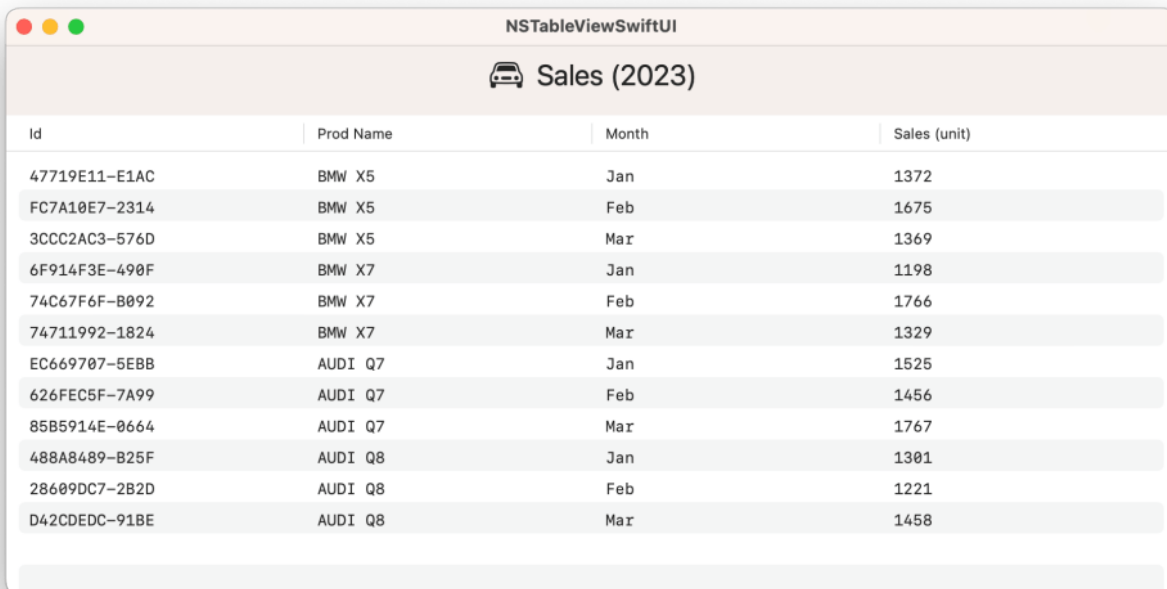


# NSTableView in SwiftUI Sample Code

 [ddas.tech/nstableview-in-swiftui-sample-code/](https://ddas.tech/nstableview-in-swiftui-sample-code/)

July 28, 2023

**NSTableView in SwiftUI** – In this post we will add a NSTableView to a SwiftUI view. Alternatively we could have used a List View from SwiftUI but NSTableView comes with a lot of features and is optimized for rendered high volume of data on the UI. This post will help mix the best of SwiftUI features and AppKit components to create a Mac OS application rich in features and capabilities



Id	Prod Name	Month	Sales (unit)
47719E11-E1AC	BMW X5	Jan	1372
FC7A10E7-2314	BMW X5	Feb	1675
3CCC2AC3-576D	BMW X5	Mar	1369
6F914F3E-490F	BMW X7	Jan	1198
74C67F6F-B092	BMW X7	Feb	1766
74711992-1824	BMW X7	Mar	1329
EC669707-5EBB	AUDI Q7	Jan	1525
626FEC5F-7A99	AUDI Q7	Feb	1456
85B5914E-0664	AUDI Q7	Mar	1767
488A8489-B25F	AUDI Q8	Jan	1301
28609DC7-2B2D	AUDI Q8	Feb	1221
D42CDEDC-91BE	AUDI Q8	Mar	1458

Lets start with creating the model that will represent each record of data for our tableview

```
struct ProductSalesRecord: Identifiable{
    var prodName: String
    var month: String
    var unitSales: Int
    var id = UUID()
    init(prodName: String, month: String, unitSales: Int) {
        self.prodName = prodName
        self.month = month
        self.unitSales = unitSales
    }
}
```

NSViewRepresentable is a wrapper that we can use to integrate an AppKit View into our SwiftUI view hierarchy.

We use `NSViewRepresentable` instance to

- Create and Manage an `NSView` object in our SwiftUI interface.
- We adopt this protocol in our app's custom interface to create, update and tear down the view.
- To add our view to the SwiftUI Interface we create our `NSViewRepresentable` instance and add it to our SwiftUI Interface, the system calls the methods of our representable instance at appropriate times to create and update the view.

**Here's a quick summary of how to use `NSViewRepresentable`:**

1. Create an `NSView` subclass: Start by creating a custom `NSView` class that encapsulates the behavior and appearance we want to display in your SwiftUI app.
2. **Adopt `NSViewRepresentable`:** Make our `NSView` subclass conform to the `NSViewRepresentable` protocol. This protocol also has two associated types: `NSViewType` and `Context`. We will need to specify the actual `NSView` type and a context type that conforms to the `NSViewRepresentableContext` protocol.
3. Implement the required methods: `NSViewRepresentable` requires us to implement two methods:
  - `makeNSView(context:)`: This method should create and return an instance of your custom `NSView`.
  - `updateNSView(_:context:)`: In this method, we update the `NSView` with the latest SwiftUI configuration and data.
4. Use the `NSViewRepresentable` in SwiftUI: Once we have created the `NSViewRepresentable`, we can use it as a SwiftUI view in our app, similar to any other SwiftUI view.

```

struct SwiftUINSTableView: NSViewRepresentable{
    @Binding var tableViewHeaders: [String]
    @Binding var tableViewData: [ProductSalesRecord]

    class Coordinator: NSObject, NSTableViewDelegate, NSTableViewDataSource {
        let tableViewHeaders: [String]
        let tableViewData: [ProductSalesRecord]

        init (tableViewHeaders:[String],tableViewData:[ProductSalesRecord]){
            self.tableViewHeaders = tableViewHeaders
            self.tableViewData = tableViewData
        }

        func numberOfRows(in tableView: NSTableView) -> Int {
            tableViewData.count
        }

        func tableView(_ tableView: NSTableView, viewFor tableColumn: NSTableColumn?,
row: Int) -> NSView? {
            let dataRow = tableViewData[row]
            var colValue : String
            if let identifier = tableColumn?.identifier.rawValue{
                switch identifier{
                    case "Prod Name":
                        colValue = dataRow.prodName
                    case "Month":
                        colValue = dataRow.month
                    case "Sales (unit)":
                        colValue = "\(dataRow.unitSales)"
                    case "Id":
                        colValue = "\(dataRow.id)"
                    default:
                        colValue = identifier
                }
            }
            let text = NSTextField(labelWithString: colValue)
            text.font = .monospacedSystemFont(ofSize: 12, weight: .regular)
            text.frame = .init(x: 0, y: 3, width: 100.0, height: 15.0)
            let cell = NSTableCellView()
            cell.addSubview(text)
            return cell
        }
        return nil
    }
}

func makeCoordinator() -> Coordinator {
    Coordinator(tableViewHeaders: tableViewHeaders, tableViewData: tableViewData)
}

func makeNSView(context: Context) -> NSScrollView {
    let tableView = NSTableView()
    tableView.usesAlternatingRowBackgroundColors = true

```

```

tableView.delegate = context.coordinator
tableView.dataSource = context.coordinator

for item in tableViewHeaders{
    let column = NSTableColumn(identifier: .init(item))
    column.title = item
    column.resizingMask = .autoresizingMask
    column.width = 100
    column.minWidth = 10
    column.maxWidth = 400
    column.headerCell.alignment = .left
    column.resizingMask = .autoresizingMask
    tableView.addTableColumn(column)
}
tableView.columnAutoresizingStyle = .uniformColumnAutoresizingStyle
let scrollView = NSScrollView()
scrollView.documentView = tableView
return scrollView
}

func updateNSView(_ nsView: NSScrollView, context: Context) {
}
}

```

Finally we will implement the ContentView to create some sample data for our Sales Record and headers for our table view columns.

In the body you can see that we are mixing a SwiftUI Label and a NSTableView NSViewRepresentable object.

```

struct ContentView: View {
    @State var headers = ["Id", "Prod Name", "Month", "Sales (unit)"]
    @State var salesData: [ProductSalesRecord] = {
        var data:[ProductSalesRecord] = []
        let prodNames = ["BMW X5", "BMW X7", "AUDI Q7", "AUDI Q8"]
        let upperBound = 2000
        let lowerBound = 1000
        for prodName in prodNames {
            for month in ["Jan", "Feb", "Mar"]{
                let rec = ProductSalesRecord(prodName: prodName, month: month,
unitSales: Int(arc4random_uniform(UInt32(upperBound - lowerBound))) + lowerBound)
                data.append(rec)
            }
        }
        return data
    }()

    var body: some View {
        Label("Sales (2023)", systemImage: "car").font(.title).padding([.top,
.bottom], 10)
        SwiftUI NSTableView(tableViewHeaders: $headers, tableViewData: $salesData)
    }
}

```

You can download the code from the below Url

[NSTableViewSwiftUIDownload](#)

You can also read the below post where we have added a NSViewController in SwiftUI using **NSViewControllerRepresentable**

| [NSViewController in SwiftUI](#)