Apple Developer Academy ✦ POA

SwiftData

Persiste data in a easy way! 🏦 de 🎲

Eduardo Riboli, Developer Mentor

History

CoreData

SwiftData

CloudKit

Evolution

CoreData

SwiftData

```
@Model

@Query

@Environment
```

@Query

@Environment

@Model

First of all we need to add the **@Model** Macro and we cannot work with Struct, we have to work with classes.

How we use: @Model


```
struct User: Identifiable {
    var id: UUID
    var name: String
    var password: String
    var isAdmin: Bool
}
```

How we use: @Model


```
final class User: Identifiable {
    var id: UUID = UUID()
    var name: String
    var password: String
    var isAdmin: Bool

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

How we use: @Model

This is how we create a Model in SwiftData

```
import SwiftData

@Model
final class User: Identifiable {
    var id: UUID = UUID()
    var name: String
    var password: String
    var isAdmin: Bool

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

How we use: @Model


```
import SwiftUI
import SwiftData

@main
struct TasksApp: App {
    var body: some Scene {
        WindowGroup {
            TabBar()
        }
        .modelContainer(for: User.self)
    }
}
```

How we use: @Model


```
import SwiftUI
import SwiftData

@main
struct TasksApp: App {
    var body: some Scene {
        WindowGroup {
            TabBar()
        }
        .modelContainer(for: [User.self, Pet.self])
    }
}
```

@Model

SwiftData

USER

id	name	password	isAdmin

@Query

@Environment

@Model

First of all we need to add the **@Model** Macro and we cannot work with Struct, we have to work with classes.

@Model

@Environment

@Query

@Query macro for querying model objects from a SwiftUI view, optionally providing a sort order or a filter predicate.

How we use: @Query


```swift
import SwiftUI

struct ContentView: View {
    @State private var users: [User] = []
    @State private var text: String = ""
    @State private var showDetails: Bool = false

    var body: some View {
        List(users) { user in
            UserDetails(user: user)
                .onTapGesture {
                    showDetails = true
                }
        }
                .searchable(text: $text)
        .sheet(isPresented: $showDetails) {
            DetailsView()
        }
    }
}
```

How we use: @Query


```swift
import SwiftUI

struct ContentView: View {
    @State private var users: [User] = []
    @State private var text: String = ""
    @State private var showDetails: Bool = false

    var body: some View {
        List(users) { user in
            UserDetails(user: user)
                .onTapGesture {
                    showDetails = true
                }
        }
        .searchable(text: $text)
        .sheet(isPresented: $showDetails) {
            DetailsView()
        }
    }
}
```

How we use: @Query


```swift
import SwiftUI
import SwiftData

struct ContentView: View {
    @Query private var users: [User]
    @State private var text: String = ""
    @State private var showDetails: Bool = false

    var body: some View {
        List(users) { user in
            UserDetails(user: user)
                .onTapGesture {
                    showDetails = true
                }
        }
                .searchable(text: $text)
        .sheet(isPresented: $showDetails) {
            DetailsView()
        }
    }
}
```

How we use: `@Query` with predicate


```swift
import SwiftUI
import SwiftData

struct ContentView: View {
    @Query(filter: #Predicate<User> { user in
        user.isAdmin == true
    }) private var users: [User]
    @State private var text: String = ""
    @State private var showDetails: Bool = false

    var body: some View {
        List(users) { user in
            UserDetails(user: user)
                .onTapGesture {
                    showDetails = true
                }
        }
                .searchable(text: $text)
        .sheet(isPresented: $showDetails) {
            DetailsView()
        }
    }
}
```

How we use: `@Query` with sort


```swift
import SwiftUI
import SwiftData

struct ContentView: View {
    @Query(sort: \User.name) private var users: [User]
    @State private var text: String = ""
    @State private var showDetails: Bool = false

    var body: some View {
        List(users) { user in
            UserDetails(user: user)
                .onTapGesture {
                    showDetails = true
                }
        }
                .searchable(text: $text)
        .sheet(isPresented: $showDetails) {
            DetailsView()
        }
    }
}
```

How we use: `@Query` with sort


```swift
import SwiftUI
import SwiftData

struct ContentView: View {
    @Query(sort: \User.name, order: .reverse) private var users: [User]
    @State private var text: String = ""
    @State private var showDetails: Bool = false

    var body: some View {
        List(users) { user in
            UserDetails(user: user)
                .onTapGesture {
                    showDetails = true
                }
        }
        .searchable(text: $text)
        .sheet(isPresented: $showDetails) {
            DetailsView()
        }
    }
}
```

How we use: @Query


```
@Query private var users: [User]
```

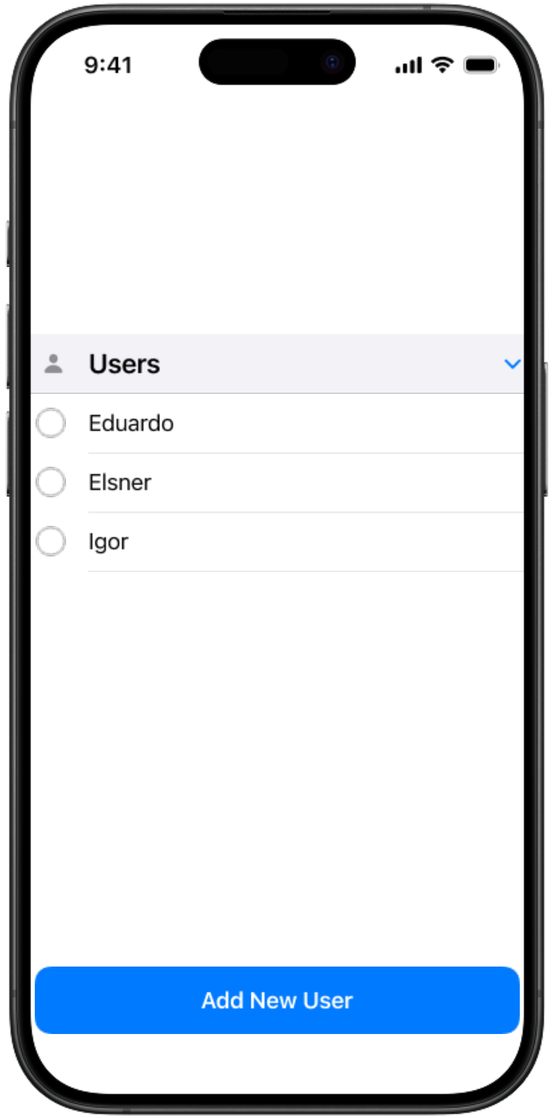
1°

2°

USER

id	name	password	isAdmin
1	Riboli	Riboli123	FALSE
2	Elsner	DoubleElsner	TRUE
3	Igor	Igão027	TRUE

SwiftData



@Model

@Environment

@Query

@Query macro for querying model objects from a SwiftUI view, optionally providing a sort order or a filter predicate.

@Model

@Query

@Environment

@Environment macro is required when you want to access variable of the system, so we will use that to get the modelContext of SwiftData.

How we use: @Environment


```
import SwiftUI

struct AddUser: View {

    @Environment(\.modelContext) var modelContext

    var body: some View {

        Button("Add User") {
            var user: User = User(name: "Dharana", password:
"Design123", isAdmin: true)

            modelContext.insert(user)
        }

    }
. . .
```

How we use: @Environment


```
import SwiftUI

struct AddUser: View {

    @Environment(\.modelContext) var modelContext

    var body: some View {

        Button("Add User") {
            var user: User = User(name: "Dharana", password:
"Design123", isAdmin: true)

            modelContext.insert(user)
        }

    }
. . .
```

How we use: `@Environment`

#Academy

`modelContext.insert(user)`

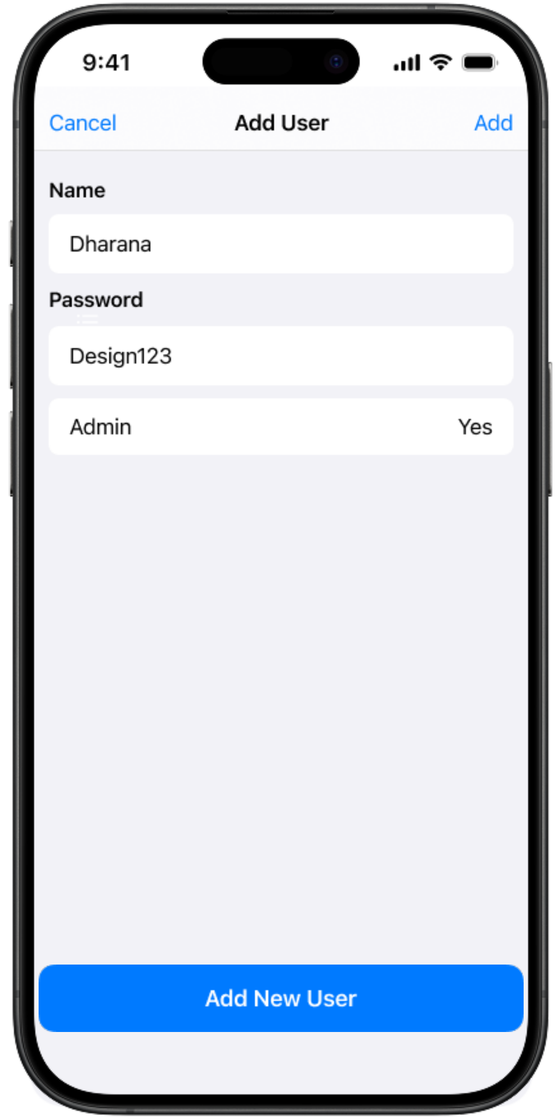
USER

id	name	password	isAdmin
1	Riboli	Riboli123	FALSE
2	Elsner	DoubleElsner	TRUE
3	Igor	Igão027	TRUE

SwiftData

2°

1°



How we use: `@Environment`

#Academy

`modelContext.insert(user)`

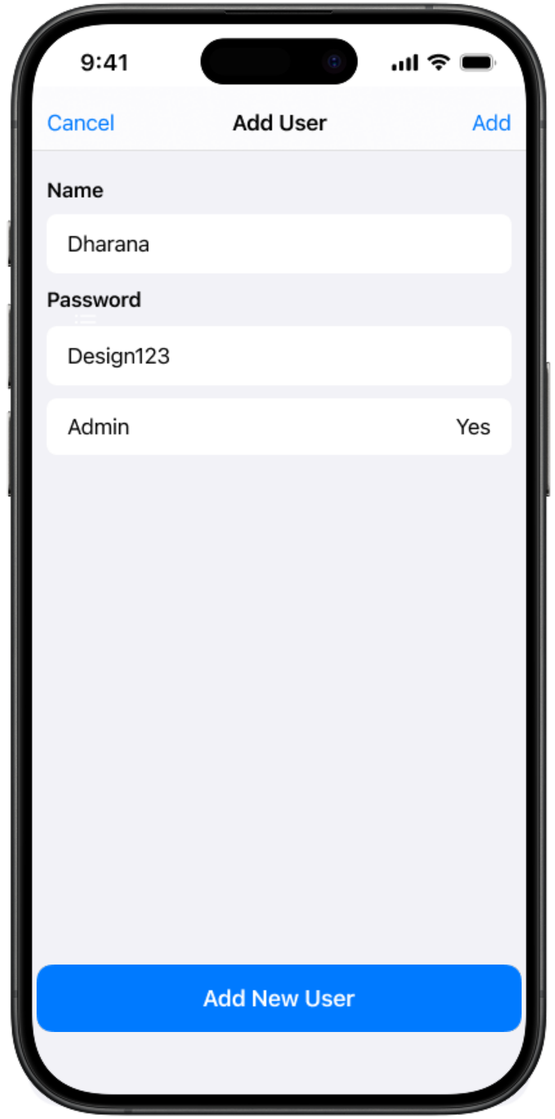
USER

id	name	password	isAdmin
1	Riboli	Riboli123	FALSE
2	Elsner	DoubleElsner	TRUE
3	Igor	Igão027	TRUE
4	Dharana	Design123	TRUE

2°

1°

SwiftData



How we use: `@Environment`


```
modelContext.delete(user)
```

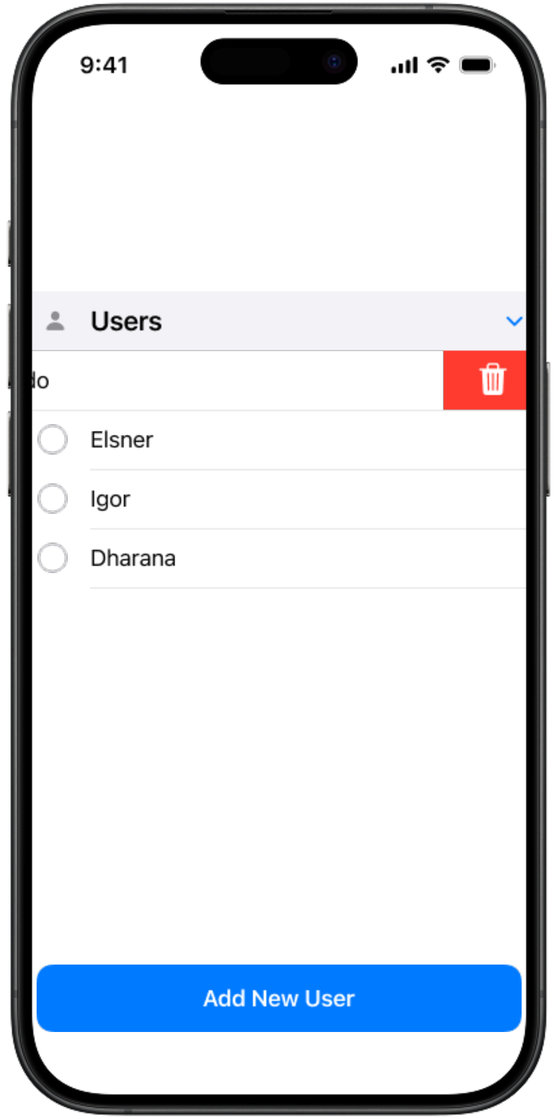
1°

2°

USER

id	name	password	isAdmin
1	Riboli	Riboli123	FALSE
2	Elsner	DoubleElsner	TRUE
3	Igor	Igão027	TRUE
4	Dharana	Design123	TRUE

SwiftData



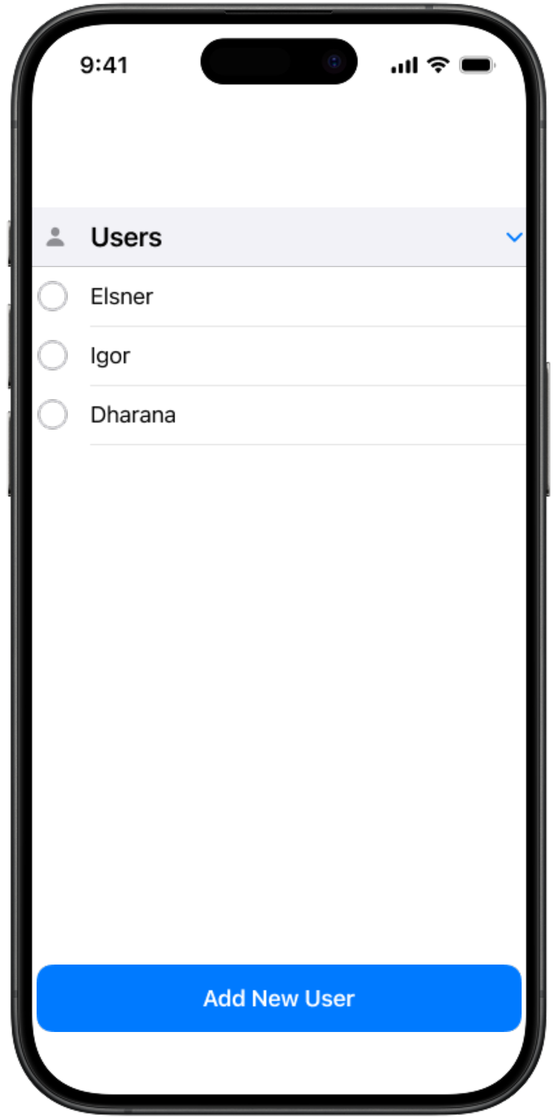
How we use: `@Environment`

#Academy

```
modelContext.delete(user)
```

1°

2°



USER

id	name	password	isAdmin
2	Elsner	DoubleElsner	TRUE
3	Igor	Igão027	TRUE
4	Dharana	Design123	TRUE

SwiftData

How we use: @Environment

Only change the property value

```
...

    HStack {
        Image(systemName: isAdmin ? "checkmark.circle.fill" : "circle")
            .foregroundStyle(isAdmin ? .accent : .gray3)
            .frame(width: 22, height: 22)

        Text("Admin")
            .padding()
    }
    .onTapGesture {
        isAdmin.toggle()
    }

...
```

How we use: **Only change the property value**

```
...

    HStack {
        Image(systemName: isAdmin ? "checkmark.circle.fill" : "circle")
            .foregroundStyle(isAdmin ? .accent : .gray3)
            .frame(width: 22, height: 22)

        Text("Admin")
            .padding()
    }
    .onTapGesture {
        isAdmin.toggle()
    }

...
```

toggle

USER

id	name	password	isAdmin
2	Elsner	DoubleElsner	TRUE
3	Igor	Igão027	TRUE
4	Dharana	Design123	TRUE

SwiftData

How we use: **Only change the property value**


```
...

    HStack {
        Image(systemName: isAdmin ? "checkmark.circle.fill" : "circle")
            .foregroundStyle(isAdmin ? .accent : .gray3)
            .frame(width: 22, height: 22)

        Text("Admin")
            .padding()
    }
    .onTapGesture {
        isAdmin.toggle()
    }

...
```

toggle

USER

id	name	password	isAdmin
2	Elsner	DoubleElsner	FALSE
3	Igor	Igão027	TRUE
4	Dharana	Design123	TRUE

SwiftData


DESIGNED BY APPLE IN CALIFORNIA