SwiftUI / Environment

Structure

Environment

A property wrapper that reads a value from a view’s environment.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 13.0+ | visionOS 1.0+ | watchOS 6.0+

```
@frozen @propertyWrapper
struct Environment<Value>
```

Mentioned in

- Building and customizing the menu bar with SwiftUI
- Managing search interface activation
- Migrating to the SwiftUI life cycle

Overview

Use the `Environment` property wrapper to read a value stored in a view’s environment. Indicate the value to read using an `EnvironmentValues` key path in the property declaration. For example, you can create a property that reads the color scheme of the current view using the key path of the `colorScheme` property:

```
@Environment(\.colorScheme) var colorScheme: ColorScheme
```

You can condition a view’s content on the associated value, which you read from the declared property’s `wrappedValue`. As with any property wrapper, you access the wrapped value by directly referring to the property:

```
if colorScheme == .dark { // Checks the wrapped value.
    DarkContent()
} else {
    LightContent()
}
```

If the value changes, SwiftUI updates any parts of your view that depend on the value. For example, that might happen in the above example if the user changes the Appearance settings.

You can use this property wrapper to read — but not set — an environment value. SwiftUI updates some environment values automatically based on system settings and provides reasonable defaults for others. You can override some of these, as well as set custom environment values that you define, using the `environment(_:_:)` view modifier.

For the complete list of environment values SwiftUI provides, see the properties of the `EnvironmentValues` structure. For information about creating custom environment values, see the `Entry()` macro.

Get an observable object

You can also use `Environment` to get an observable object from a view's environment. The observable object must conform to the `Observable` protocol, and your app must set the object in the environment using the object itself or a key path.

To set the object in the environment using the object itself, use the `environment(_:)` modifier:

```
@Observable
class Library {
    var books: [Book] = [Book(), Book(), Book()]

    var availableBooksCount: Int {
        books.filter(\.isAvailable).count
    }
}

@main
struct BookReaderApp: App {
    @State private var library = Library()

    var body: some Scene {
        WindowGroup {
            LibraryView()
```

```
                    .environment(library)
            }
        }
```

To get the observable object using its type, create a property and provide the `Environment` property wrapper the object's type:

```
struct LibraryView: View {
    @Environment(Library.self) private var library

    var body: some View {
        // ...
    }
}
```

By default, reading an object from the environment returns a non-optional object when using the object type as the key. This default behavior assumes that a view in the current hierarchy previously stored a non-optional instance of the type using the `environment(_:)` modifier. If a view attempts to retrieve an object using its type and that object isn't in the environment, SwiftUI throws an exception.

In cases where there is no guarantee that an object is in the environment, retrieve an optional version of the object as shown in the following code. If the object isn't available the environment, SwiftUI returns `nil` instead of throwing an exception.

```
@Environment(Library.self) private var library: Library?
```

Get an observable object using a key path

To set the object with a key path, use the `environment(_:_:)` modifier:

```
@Observable
class Library {
    var books: [Book] = [Book(), Book(), Book()]

    var availableBooksCount: Int {
        books.filter(\.isAvailable).count
    }
}
```

```swift
@main
struct BookReaderApp: App {
    @State private var library = Library()

    var body: some Scene {
        WindowGroup {
            LibraryView()
                .environment(\.library, library)
        }
    }
```

To get the object, create a property and specify the key path:

```swift
struct LibraryView: View {
    @Environment(\.library) private var library

    var body: some View {
        // ...
    }
}
```

Topics

Creating an environment instance

`init(_:)`

Creates an environment property to read the specified key path.

Getting the value

`var wrappedValue: Value`

The current value of the environment property.

Relationships

Conforms To

`DynamicProperty`, `Sendable`, `SendableMetatype`

See Also

Accessing environment values

`struct EnvironmentValues`

A collection of environment values propagated through a view hierarchy.