< What is the @ObservedObject property wrapper?

What is the @Environment property wrapper? >

What is the @EnvironmentObject property wrapper?

Paul Hudson @twostraws February 9th 2021

Updated for Xcode 13.2

SwiftUI's `@EnvironmentObject` property wrapper lets us create views that rely on shared data, often across an entire SwiftUI app. For example, if you create a user that will be shared across many parts of your app, you should use `@EnvironmentObject`.

For example, we might have an `Order` class like this one:

```
class Order: ObservableObject {
    @Published var items = [String]()
}
```

That conforms to `ObservableObject`, which means we can use it with either `@ObservedObject` or `@EnvironmentObject`. In this instance, we might create a view that uses it with `@EnvironmentObject`, like this:

```
struct ContentView: View {
    @EnvironmentObject var order: Order

    var body: some View {
        // your code here
    }
}
```

Notice how the `order` property isn't given a default value – by using `@EnvironmentObject` we're saying that value will be provided by the SwiftUI environment rather than explicitly created by this view.

`@EnvironmentObject` has a lot in common with `@ObservedObject`: both must refer to a class that conforms to `ObservableObject`, both can be shared across many views, and both will update any views that are watching when significant changes happen. However, `@EnvironmentObject` specifically means "this object will be provided from some outside entity, rather than being created by the current view or specifically passed in.

In practical terms, imagine if you had view A, and view A had some data that view E wanted. Using `@ObservedObject` view A would need to hand the object to view B, which would hand it to view C, then view D, and finally view E – all the intermediate views would need to be sent the object even though they didn't actually need it.

When using `@EnvironmentObject`, view A can create an object and place it into the environment. Any views inside it can then gain access to that environment object whenever they want just by asking for it, rather than having to pass it around explicitly – it makes our code much simpler.

Warning: When a view using `@EnvironmentObject` is shown, SwiftUI will immediately search the environment for an object of the correct type. If such an object can't be found – i.e., if you forgot to place it in the environment – then your app will immediately crash. When you use `@EnvironmentObject` you are effectively promising that object will exist in the environment by the time it is needed, a bit like using implicitly unwrapped optionals.

Similar solutions...

- How to use @EnvironmentObject to share data between views
- What's the difference between @ObservedObject, @State, and @EnvironmentObject?
- Adding items to an order with @EnvironmentObject
- All SwiftUI property wrappers explained and compared
- What is the @GestureState property wrapper?

< What is the @ObservedObject property wrapper?

What is the @Environment property wrapper? >